Sérgio Vasconcelos Castro



Online Order Allocation: From Problem Classification to Decision Support

Submitted to Faculdade de Engenharia da Universidade do Porto in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Industrial Engineering and Management, supervised by Gonçalo Figueira, Associate Professor of Faculdade de Engenharia da Universidade do Porto, and Bernardo Almada-Lobo, Full Professor of Faculdade de Engenharia da Universidade do Porto

Department of Industrial Engineering and Management
Faculdade de Engenharia da Universidade do Porto
2026

This research was partially supported by the PhD grant 2021.05488.BD awarded by the Portuguese Foundation for Science and Technology and by the European Union's Horizon 2020 research and innovation programme under grant agreement No 952060 (TRUST-AI).

Bridegroom, dear to my heart,
Goodly is your beauty, honey-sweet,
Lion, dear to my heart,
Goodly is your beauty, honey-sweet.

Istanbul #2461
Sumerian love poem, c. 2037–2029 BCE

Acknowledgments

I want to start by thanking my wife for being my companion throughout my PhD journey. We laughed, cried, traveled, worked, and loved each other throughout these years. Thank you for your patience and for providing the support necessary to do this work.

A word of appreciation and acknowledgment goes to my family. To all my uncles, aunts, cousins, and grandmothers. A special thank you to my closest family: my father, my sister, and my nephew, who have supported and given great joy during this journey; and to my mother, who, although no longer with us, had an unquestionable impact on my education and upbringing, and who I have not stopped loving.

I also want to thank my friends from my high school years, Paulo, Vasco, Diogo, Artur, Capitão, and Tomé; and my bachelor's and master's friends, Morim, Rui, Ana Coelho, Ana Carvalho, Joana, Rafa, and Beatriz. Hanging out with you was great fun, a pleasure, and overall amazing. I could not do it without your support. Also, a word for my Erasmus friends, especially Dionysia, Marianna, Vasiliki, and Melissa: although we live across Europe, our friendship endures, and visiting each other is something that I always look forward to.

Next, I would like to thank everyone who has supported me in these years at FEUP and INESC TEC. Not just for the feedback and support that is naturally exchanged, but also, and mostly, for the friendship we have developed in the process. A special thanks to the IEMS and DEGI Club organization team I worked with, especially Marcella, who brought excellence to those scientific and networking events. Finally, to all the people who, during these years, worked at SYSTEM/former CEGI at INESC TEC, especially Sara Ali, Massoud, Luciana Yamada, Felipe, Mariana, Daniela, Miguel, Ângelo, José, Rocha, João, Mário, Fábio, Ricardo, Sara Martins, Maria João, Sara Neves, Flávia, António Cardoso, António Almeida, and to visiting researchers Christoph, Luciana Casacio, Paula, Simone, Igor, and Laura. There are two people that I would like to give a special shout-out: Francisco, with whom I worked closely on an important project and on multiple research proposals, always providing support beyond expectations; and Nuno, who is my "twin" in this PhD journey and shared with me all the pains of the process and FEUP/INESC's gossip.

Special gratitude goes to the FEUP, FCT, and INESC TEC institutions, which have supported my studies through scholarships, support for presenting my work at national and international conferences, such as the EURO and IO conferences, and my short stay at TUE. On the latter, I also want to thank TUE and everyone from the OPAC department with whom I worked and who helped elevate my research. A few people I met during my stay in Eindhoven, and would like to give a special acknowledgment to are: Federico and Clement, my roommates with whom I hung out, Miguel, with whom I especially enjoyed my time doing sports, and most importantly Willem, who was the person who received me, supervised my work, and helped me network with the people at TUE and the dutch operations research community in general.

To conclude, I want to give a special thanks to my supervisors, Bernardo and Gonçalo, for their patience, feedback, and for keeping pushing me forward, especially in the most difficult times. Thank you for working with me. I hope you remember our time together

fondly and, above all, that you are proud of our work.

Abstract

Online retail has transformed fulfillment from a process in which customers selected the source of supply, by simply walking into a store, into a retailer-directed decision. In this setting, incoming online demand must be dynamically assigned to available inventory across a network of fulfillment nodes. This decision is especially relevant in online marketplaces, where fulfillment sources may involve third-party vendors, potentially affecting vendor satisfaction as well as customer satisfaction and fulfillment costs.

This thesis examines online allocation and order routing, with particular emphasis on online marketplaces. Its overarching objective is to investigate how order allocation can be structured, modeled, evaluated, and supported under realistic operational conditions. The thesis addresses this objective through three complementary perspectives: first, by structuring and classifying the literature on online order allocation; second, by extending the problem to marketplace-specific vendor satisfaction objectives; and third, by developing a data-driven decision-support framework for evaluating and improving deployable allocation policies.

The first contribution is conceptual. The thesis defines online order allocation as a distinct operational decision within online retail fulfillment and organizes a fragmented literature around a baseline problem, problem-extension dimensions, and solution-method dimensions. This classification clarifies the relationship between order allocation and adjacent fulfillment decisions and supports comparisons across problem variants. It also identifies research opportunities in underexplored problem variants, multi-objective optimization, reinforcement learning, and broader benchmarking of fulfillment policies.

The second contribution concerns marketplace modeling. The thesis extends the multi-item order fulfillment problem to online marketplaces. In this model, the platform allocates demand across vendor-owned inventory while considering fulfillment costs, unfulfilled demand, and vendor satisfaction. Vendor satisfaction is represented by allocation targets and shortfall penalties. Computational experiments show that vendor satisfaction creates a meaningful trade-off between fulfillment costs and target shortfalls: satisfaction-aware policies can reduce target shortfalls, but often at the expense of higher fulfillment costs. The comparison of myopic, cost-function, randomized, and learning-based policies shows that no single policy class dominates across all settings.

The third contribution is methodological and practical. The thesis proposes a modular, data-driven, simulation-based decision-support framework for evaluating and improving allocation policies before deployment. The framework transforms operational data into simulation instances, supports policy evaluation and robustness analysis, and uses genetic programming to optimize interpretable allocation policies. A real-world fashion marketplace case study illustrates how operational data can be used to improve fulfillment policies while preserving compatibility with existing business logic. Overall, the thesis advances the view of order allocation beyond cost minimization, positioning it as a multi-stakeholder and deployable decision-support problem.

Resumo

O retalho *online* transformou a satisfação da procura, de um processo em que os clientes selecionam o local de abastecimento ao entrarem numa loja, para uma decisão do retalhista. Neste contexto, a procura *online* deve ser alocada dinamicamente ao inventário disponível numa rede de abastecimento. Esta decisão é especialmente relevante em *marketplaces* online, onde as fontes de abastecimento podem corresponder a vendedores externos e as decisões de alocação podem afetar não só os custos de entrega e a qualidade do serviço, mas também a forma como as vendas são distribuídas entre os *stakeholders* do *marketplace*.

Esta tese estuda a alocação de encomendas no retalho *online*, com ênfase particular nos *marketplaces online*. O seu objetivo principal é analisar como a alocação de encomendas pode ser estruturada, modelada, avaliada e apoiada em condições operacionais realistas. A tese aborda este objetivo através de três perspetivas complementares: primeiro, estruturando e classificando a literatura sobre *online order allocation*; segundo, estendendo o problema através da incoportação de objetivos de satisfação dos vendedores, específicos de *marketplaces*; e, terceiro, desenvolvendo uma *framework* de apoio à decisão, que utiliza dados para avaliar e melhorar políticas de alocação.

A primeira contribuição é conceptual. A tese define *online order allocation* como uma decisão operacional distinta, no âmbito da entrega de encomendas no retalho *online*, e organiza a literatura fragmentada em torno de um problema-base, com a sua extensão através de várias dimensões e a classificação dos métodos de solução. Esta classificação clarifica a relação entre a alocação de encomendas e outras decisões operacionais, apoia a comparação entre variantes do problema e identifica oportunidades de investigação em variantes ainda pouco exploradas, métodos de otimização multiobjetivo e baseados em aprendizagem por reforço, e uma comparação mais abrangente entre políticas de alocação.

A segunda contribuição consiste na modelação da alocação de encomendas em *marketplaces*. A tese estende o problema de alocação de encomendas *multi-item* a *marketplaces online*, introduzindo o *multi-item order fulfillment problem in online marketplaces*. Neste modelo, a plataforma aloca a procura ao inventário detido pelos vendedores, considerando simultaneamente os custos de entrega, a procura não satisfeita e a satisfação dos vendedores. Esta última é modelada através de metas de alocação e de penalizações pelo incumprimento das mesmas. As experiências computacionais mostram que a satisfação dos vendedores gera um trade-off relevante entre custos de entrega e o cumprimento da metas: políticas sensíveis à satisfação dos vendedores podem melhorar o cumprimento das metas, mas (normalmente) à custa de custos de cumprimento mais elevados. A comparação entre políticas míopes, baseadas em funções de custo, aleatorizadas e baseadas em aprendizagem mostra ainda que nenhuma classe de políticas domina em todos os contextos.

A terceira contribuição é metodológica e prática. A tese propõe uma *framework* modular, que utiliza dados de negócio que são preparados para suportar simulações para avaliar e melhorar políticas de alocação antes da sua implementação. A *framework* transforma os dados em instâncias de simulação, sendo que as mesmas permitem tanto apoiar a avaliação de políticas e a análise da sua robustez como utilizar Programação Genética para otimizar políticas de alocação interpretáveis. Um caso de estudo real num marketplace de moda ilustra como os dados operacionais podem ser utilizados de forma prescritiva para melhorar as

políticas de alocação de encomendas, preservando a interpretabilidade e a compatibilidade com a lógica operacional existente. No global, a tese representa um avanço na forma como a alocação de encomendas deve ser abordada, indo para além da minimização de custos e posicionando-a como um problema de apoio à decisão multi-*stakeholder* e implementável.

Contents

1 Introduction **1**
- 1.1 Order allocation problem's scope and research gaps . . . 2
- 1.2 Research objectives and research questions . . . 3
- 1.3 Thesis structure and contributions . . . 4
- Bibliography . . . 8

2 Order Allocation in Online Retail: Classification and Review **11**
- 2.1 Introduction . . . 11
- 2.2 Article collection procedure . . . 15
- 2.3 Problem Characterization . . . 17
  - 2.3.1 Baseline Problem . . . 17
  - 2.3.2 Problem Extensions . . . 24
  - 2.3.3 Literature Classification and Review . . . 30
- 2.4 Allocation Policy . . . 35
  - 2.4.1 Optimization mechanism . . . 35
  - 2.4.2 Decision-making policy class and approximation/solution technique 38
  - 2.4.3 Future demand information and learning . . . 39
  - 2.4.4 Literature Classification and Discussion . . . 40
- 2.5 Research Opportunities . . . 42
  - 2.5.1 Problem settings . . . 42
  - 2.5.2 Solution Methods . . . 43
- 2.6 Conclusion . . . 44
- Bibliography . . . 45

3 Order Allocation in Online Marketplaces **55**
- 3.1 Introduction . . . 56
- 3.2 Literature . . . 58
  - 3.2.1 Fulfillment Optimization . . . 58
  - 3.2.2 Data-driven methods in inventory management . . . 59
  - 3.2.3 Online marketplaces and platform-mediated ranking . . . 60
- 3.3 Problem Setting and Model . . . 60
  - 3.3.1 Marketplace Order Fulfillment with Vendor Targets . . . 61
  - 3.3.2 MDP-EI Formulation . . . 63
- 3.4 Methods . . . 66
  - 3.4.1 Myopic policies . . . 67
  - 3.4.2 Cost-function policies . . . 68
  - 3.4.3 Dilate rounding policy . . . 69
  - 3.4.4 DCL . . . 70
- 3.5 Computational Experiments . . . 71
  - 3.5.1 Experimental motivation . . . 71

3.5.2 Instance generation . . . 72
3.5.3 Reference scenario . . . 72
3.5.4 Performance measures . . . 73
3.5.5 Vendor-target stress test . . . 75
3.5.6 Structural scenario analysis . . . 80
3.6 Conclusion . . . 82
3.A Procedural view of the model . . . 85
3.B Cost-function policy details . . . 86
3.B.1 $DLP^{CF\text{-}LP}$ . . . 86
3.B.2 CF-PD update rule . . . 88
3.C Dilate rounding policy details . . . 90
3.C.1 DLP^{R} . . . 90
3.C.2 Dilate rounding . . . 92
3.C.3 Parameterization in the computational experiments . . . 92
3.D DCL Implementation Details . . . 94
3.D.1 Algorithmic overview . . . 94
3.D.2 State features . . . 95
3.D.3 Sampling and training parameters . . . 96
3.E Detailed instance generation procedure . . . 99
3.E.1 Network and fulfillment-cost construction . . . 99
3.E.2 Item-level demand and total seasonal demand . . . 101
3.E.3 Order-arrival process . . . 101
3.E.4 Inventory initialization . . . 102
3.E.5 Allocation targets and penalty . . . 103
3.E.6 Generated instance . . . 103
3.F Policy preparation and online execution time . . . 104
Bibliography . . . 106

4 Decision Support Framework for Sequential Order Allocation 111
4.1 Introduction . . . 112
4.2 Literature Review . . . 113
4.3 Business Problem and Decision Context . . . 115
4.3.1 Problem Description and Decision Model . . . 115
4.3.2 Business Motivation and Use Cases . . . 116
4.4 Decision Support Framework for Policy Evaluation and Optimization . . . 117
4.4.1 Overview and System Architecture . . . 117
4.4.2 Scoring-Based Allocation Policies . . . 119
4.4.3 Simulation Engine . . . 120
4.4.4 GP-Based Policy Optimization . . . 121
4.5 Application to Case Study . . . 123
4.5.1 Case Study Context Description . . . 123
4.5.2 Data and Instance Construction . . . 124
4.5.3 Simulator Setup . . . 126
4.5.4 Optimization Setup . . . 127

4.5.5 Results and Discussion . . . 128
4.5.6 Practical and Managerial Implications . . . 133
4.6 Conclusion . . . 133
4.A Detailed Problem Formulation . . . 134
4.A.1 Decision Variables . . . 135
4.A.2 State Variables . . . 136
4.A.3 Exogenous Information . . . 137
4.A.4 Transition Function . . . 138
4.A.5 Objective Function . . . 139
4.A.6 Optimization Problem . . . 140
4.A.7 Model Summary . . . 140
Bibliography . . . 142

5 General discussion, conclusions, and future research 145
5.1 Answers to the research questions . . . 145
5.2 Cross-chapter synthesis . . . 148
5.3 Managerial implications . . . 151
5.4 Future research agenda . . . 153
Bibliography . . . 156

Abbreviations and Symbols

Thesis-wide

DC	Distribution Center
DCL	Deep Controlled Learning
DRL	Deep Reinforcement Learning
DSF	Decision-Support Framework
GP	Genetic Programming
MDP	Markov Decision Process
MIOFP	Multi-Item Order Fulfillment Problem
MIOFP-OM	Multi-Item Order Fulfillment Problem in Online Marketplaces
ML	Machine Learning
NN	Neural Network
OAOR	Order Allocation in Online Retail
OF	Order Fulfillment
RL	Reinforcement Learning

Chapter 2

1P	Single-Item
1U	Single-Unit
ABBS	Assigned Backlog Base-Stock
ADL	Allocation Delay
ALNS	Adaptive Large Neighbourhood Search
BOPS	Buy Online Pick Up In Store
BORIS	Buy Online Return In Store
BOSS	Buy Online Ship From Store
CFA	Cost Function Approximation
D-CO	Channel Opening/Closure
D-FR	Product Framing
D-OR	Order Rejection
D-PR	Product Pricing
D-SU	Product Substitution
DLA	Direct Lookahead Approximation
DL	Deterministic Lookahead
DM	Demand Management
DP	Dynamic Programming
F-CA	Order Cartonization
F-MS	Fulfillment Mode Selection

F-OC	Order Consolidation
F-OP	Order Picking
F-OR	Order Routing
F-TR	Reactive Inventory Transshipment
FPM	Fulfillment Process Management
GA	Genetic Algorithm
I-I	Initial Inventory
IM	Inventory Management
I-RP	Inventory Replenishment
I-TR	Proactive Inventory Transshipment
LP	Linear Program
LR	Lagrangian Relaxation
MIP	Mixed(-Integer) Program
mP	Multi-Item
MP Batch	Multi-Period Batch
mU	Multi-Unit
ObO	Order-by-Order
ODC	Omnichannel Distribution Center
ODS	Omnichannel Drop-Shipping
OVRP	Omnichannel Vehicle Routing Problem
PFA	Policy Function Approximation
RF	Randomized Fulfillment
RTT	Randomized Tabu Thresholding
SA	Static Approximation
SP Batch	Single-Period Batch
STS	Ship To Store / Buy Online Ship To Store
VFA	Value Function Approximation
VNS	Variable Neighbourhood Search
VRP	Vehicle Routing Problem

Chapter 3

CF-LP	Linear-Programming-Based Cost Function
CF-PD	Primal-Dual-Based Cost Function
DLP	Deterministic Linear Program
$\Delta FC^{\%}$	Fulfillment-cost delta (percentage)
GMV	Gross Merchandise Value
M-MI	Myopic Multi-Item policy
M-S	Myopic Satisfaction policy
MDP-EI	Markov Decision Process with Exogenous Inputs
MLP	Multilayer Perceptron
PPO	Proximal Policy Optimization

R-D	Dilate Round policy
SF^u	Remaining target shortfall (units)
$SF^{\%}$	Remaining target shortfall (percentage)
SR^u	Shortfall reduction (units)
$SR^{\%}$	Shortfall reduction (percentage)
TDR	Target-to-Demand Ratio

Chapter 4

BC	Basket Coverage
SC	Shipping Cost
SD	Stock Depth

Chapter 1

Introduction

Online retail has become a central component of modern retailing: globally, retail e-commerce is estimated to account for roughly one-fifth of all retail sales, with a projected rise from 19.9% in 2024 to 20.5% in 2025, while global online retail sales are projected to exceed US$6 trillion (Lebow, 2025; Cramer-Flood, 2024). As the commercial relevance of online retail increases, so does the importance of improving the operations that support it.

A key operational difference between traditional and online retail is illustrated in Figure 1.1. In traditional retail, customers typically visit a physical store and purchase products from the inventory available there. With this, the customer implicitly selects the fulfillment source. In contrast, in online retail, demand is captured digitally, and the retailer or platform must move products from one or more inventory-holding locations to the customer's chosen delivery location. In this context, it is the retailer who must select among several possible fulfillment nodes, such as warehouses, stores, suppliers, or third-party vendors. This makes fulfillment a central operational capability of online retail, where fulfillment becomes a retailer- or platform-directed fulfillment setting in which online demand can be routed across a network of locations that hold inventory (Agatz et al., 2008; Hübner et al., 2015; Melacini et al., 2018). The same order may therefore be fulfilled in different ways, with different implications for fulfillment cost, customer satisfaction, inventory availability, vendor or partner outcomes, and future allocation opportunities (Xu et al., 2009).

Online marketplaces are especially important within online retail. It is estimated that they generated 62% of global retail e-commerce sales in 2024, equivalent to US$2.4 trillion, while third-party sellers accounted for 81% of marketplace sales, underscoring the marketplace model's role as a core retail infrastructure (Peh, 2025). In these platforms, the marketplace connects customers and vendors, while the inventory and fulfillment capabilities may be partially or fully controlled by third parties. Multiple vendors may offer overlapping assortments, sell through external channels, and dynamically update their inventory commitments. As a result, marketplace order allocation requires selecting an appropriate fulfillment node while simultaneously coordinating among the platform, its customers, and its vendors.

This thesis is motivated by this evolution from customer-selected fulfillment in traditional retail to retailer- or platform-directed fulfillment in online retail and marketplaces. The allocation of online orders is an operational decision with strategic relevance because it links inventory availability, fulfillment cost, service performance, and stakeholder outcomes. The next section defines this decision more precisely and identifies the research gaps that the thesis's three article-based chapters address.

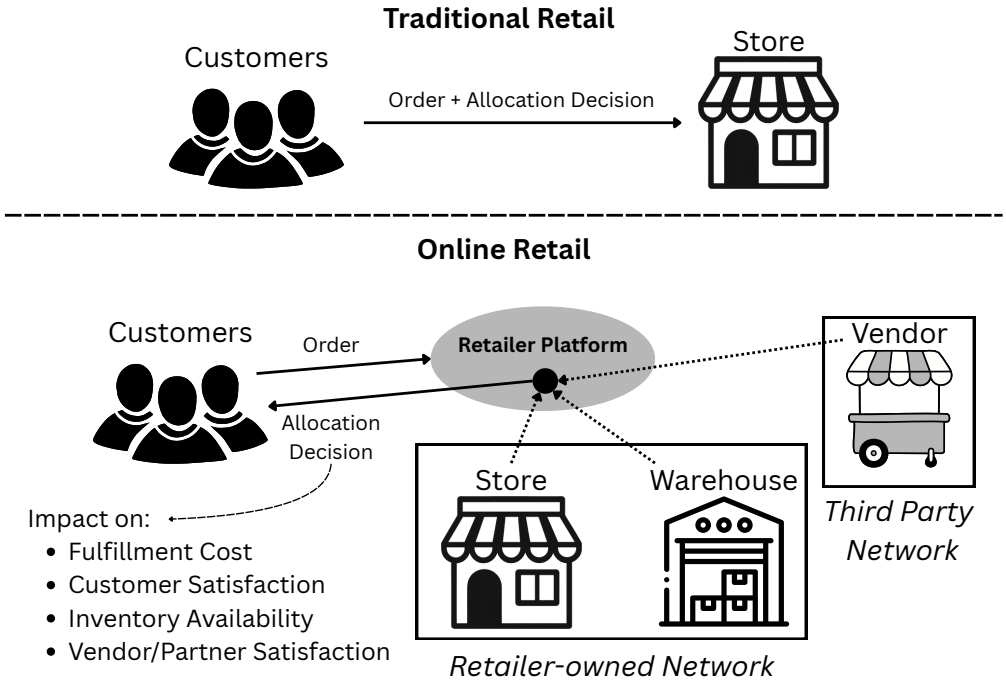


Figure 1.1: Customer-selected fulfillment in traditional retail and retailer- or platform-directed fulfillment in online retail.

1.1. Order allocation problem's scope and research gaps

The operational decision introduced in the previous section is referred to in this thesis as Order Allocation in Online Retail (OAOR). This decision involves allocating each unit of each item* in an online order to the available inventory of the same item at a fulfillment node (i.e., a store, warehouse, or another facility) that can satisfy the order's delivery constraints. This decision falls within a broader set of e-fulfillment decisions, including distribution network design, warehouse design, delivery service design, pricing, forecasting, and transportation planning. Although related to these decisions, OAOR corresponds solely to the operational assignment of online demand to fulfillment sources.

OAOR is challenging as decisions are sequential and state-dependent. Online orders arrive over time, and each allocation consumes inventory that may be useful for future orders. A decision that minimizes the immediate fulfillment cost of the current order may therefore reduce future flexibility, thereby increasing later fulfillment costs or contributing to lost sales. These trade-offs become more complex when orders contain multiple items, fulfillment nodes differ in costs and capabilities, inventory information is dynamic or uncertain, and decisions must be made quickly enough to support operational execution (Acimovic and Graves, 2015; Jasin and Sinha, 2015).

The OAOR problem, although relatively recent, has received increasing attention in

*In this thesis, we assume that an item is equivalent to a product.

the literature. Nevertheless, important gaps subsist. The first gap addressed in this thesis concerns the structure of the OAOR literature itself. As the field has grown, contributions have appeared under different names, in different retail settings, and with different modeling assumptions and solution mechanisms. This fragmentation makes it difficult to position existing work, compare problem variants, understand the range of available policy classes, and identify research opportunities.

The second gap concerns the limited treatment of marketplace-specific objectives. Existing OAOR research has largely focused on operational outcomes such as fulfillment cost, lost sales, delivery performance, and inventory availability. However, in online marketplaces, fulfillment decisions may also affect vendors, whose participation and engagement are important for the platform. When multiple vendors can supply the same product, the marketplace's allocation decisions determine not only how demand is fulfilled, but also how sales opportunities are distributed across vendors. This concern relates to the broader literature on the fairness of exposure in online allocation and ranking systems, which emphasizes that platform decisions can affect the opportunities available to the items or providers being ranked (Singh and Joachims, 2018).

The third gap concerns the practical evaluation and improvement of allocation policies. Many optimization-based approaches rely on assumptions that may be difficult to satisfy in operational systems, such as reliable demand forecasts and accurate inventory information. In practice, retailers and marketplaces often rely on heuristic or rule-based allocation policies because they are fast, interpretable, and easier to integrate with existing systems (Ye et al., 2026). However, such policies are rarely evaluated or improved systematically across alternative operating conditions.

1.2. Research objectives and research questions

Building on the research gaps identified, the objective of this thesis is to contribute to the fulfillment optimization literature in online retail by clarifying the OAOR problem space, extending it to marketplace-specific vendor objectives, and studying how allocation policies can be evaluated and improved using operational data under practical deployment requirements. Accordingly, the overarching research question guiding this thesis is the following:

> **ORQ.** How can OAOR, particularly in online marketplace settings, be structured, modeled, evaluated, and supported under realistic operational conditions?

The first research question addresses the need to clarify the problem space. Since OAOR has been studied under different names, in different retail settings, and with different modeling assumptions, a systematic characterization is necessary to position existing contributions and identify research opportunities. This leads to the first research question:

> **RQ1.** How can OAOR be systematically characterized, and what problem variants, solution approaches, and research gaps emerge from the literature?

The second research question focuses on online marketplaces. In these settings, fulfillment sources may be third-party vendors, necessitating the explicit modeling of vendor-related objectives rather than focusing solely on fulfillment costs, lost sales, or customer-facing service outcomes. The second research question is therefore:

> **RQ2.** How does the order allocation problem change in online marketplaces when vendor satisfaction is explicitly considered as a fulfillment objective?

The third research question concerns the practical support of allocation decisions. Even though sophisticated optimization methods have been proposed in literature, real retail systems often require allocation policies that are fast, interpretable, robust to operational uncertainty, and compatible with existing decision logic. This motivates the study of decision-support tools that use operational data to evaluate and improve deployable policies prior to implementation. Consequently, the third research question is:

> **RQ3.** How can order allocation policies be evaluated and improved using operational data while preserving interpretability, robustness, and deployment feasibility?

These questions define the scope of the thesis. The thesis contributes to the literature by (1) structuring the OAOR problem space, (2) developing a marketplace-specific modeling extension, and (3) proposing a configurable Decision-Support Framework (DSF) for policy evaluation and improvement. Across these contributions, the thesis also examines how different allocation policy representations can be classified, adapted, compared, and operationalized. Table 1.1 summarizes the correspondence between the research questions, article-based chapters, and main outputs. Section 1.3 then discusses how these chapters complement one another in the overall thesis.

1.3. Thesis structure and contributions

This thesis follows an article-based organization comprising three research papers framed by this introductory chapter and a final concluding chapter. Although each paper addresses a specific research question, together they develop a sequence of contributions that moves from structuring the OAOR field to extending the problem to marketplace-specific vendor objectives to proposing a data-driven DSF for evaluating and improving allocation policies in realistic operational settings.

Chapter 2, entitled *Order Allocation in Online Retail: Classification and Review*, provides the conceptual foundation of the thesis. It defines OAOR, formulates a baseline problem, classifies inventory management, demand management, and fulfillment process management related problem extensions, and organizes solution methods according to their

Table 1.1: Research gaps, questions, and main thesis outputs.

Research gap	Why it matters	RQ	Main output	Chap.
Fragmented OAOR literature, inconsistent terminology, and limited cross-policy benchmarking	Makes it difficult to position existing work, compare problem variants and policy classes, and identify research opportunities.	**1.**	OAOR definition, baseline problem, classification of extensions, solution-method taxonomy, and research opportunities.	**2.**
Limited modeling of marketplace-specific vendor objectives	Marketplace allocation decisions affect not only cost and service, but also how sales opportunities are distributed among vendors.	**2.**	Marketplace order fulfillment problem formulation, vendor satisfaction modeling through allocation targets and shortfall penalties, and analysis of fulfillment-cost/vendor-satisfaction trade-offs across distinct policy classes.	**3.**
Limited deployable decision-support tools for policy evaluation and improvement	Practical allocation policies must be fast, interpretable, robust, and testable before deployment in operational systems.	**3.**	Modular, data-driven, simulation-based DSF for policy evaluation and optimization using operational data, instantiated in a real-world fashion marketplace case study.	**4.**

main mechanisms, policy classes, and use of future information and learning. In the thesis's logic, this chapter establishes the vocabulary and structure needed to position the following two studies. It also identifies research opportunities in under-addressed problem variants and solution methods, including marketplace-related vendor objectives, learning-based optimization, and broader policy benchmarking. This chapter addresses RQ1.

Chapter 3, entitled *Order Allocation in Online Marketplaces*, builds on this foundation by focusing on online marketplaces. In these settings, the allocation decision is not limited to selecting a low-cost fulfillment source, since the platform may also influence how sales opportunities are distributed among vendors. The chapter, therefore, extends the classical Multi-Item Order Fulfillment Problem (MIOFP) to marketplace settings by introducing vendor satisfaction as an explicit objective, modeled through allocation targets and shortfall penalties. It then compares multiple allocation policies across distinct policy classes

and analyzes how target ambition, penalty intensity, target granularity, and marketplace structure affect the resulting fulfillment-cost/vendor-satisfaction trade-offs. This chapter addresses RQ2.

Chapter 4, entitled *Decision Support Framework for Sequential Order Allocation*, addresses the practical support of allocation decisions. It proposes a modular, data-driven, simulation-based DSF that can be instantiated for different order allocation contexts, objectives, constraints, and policy representations. In the case study, the framework is tailored to a real fashion marketplace and focuses on score-based allocation policies because they are interpretable and compatible with existing operational logic. Genetic Programming (GP) is used to optimize this policy representation from simulation-based performance feedback, following the idea that rule-like policies can be learned while preserving interpretability (Ferreira et al., 2022). Many online retailers and marketplaces share characteristics similar to those observed in the case study: a collection of historical business data, sparse item-level demand, and existing score-based allocation logic with strong requirements for interpretability, robustness, and deployment feasibility. The chapter, therefore, provides a configurable framework and a set of operational design choices that can be transferred to other businesses with similar fulfillment characteristics, addressing RQ3.

A secondary methodological theme also runs across the three article-based chapters: the organization and comparison of distinct allocation policies. Chapter 2 reviews and classifies the solution methods proposed in the OAOR literature, including their optimization mechanisms, policy classes, and use of future information and learning. Chapter 3 then adapts multiple policies from distinct policy classes to the Multi-Item Order Fulfillment Problem in Online Marketplaces (MIOFP-OM). These policy classes build on different streams of the Order Fulfillment (OF) and sequential decision-making literature, including opportunity-cost or value-function-based fulfillment policies (Acimovic and Graves, 2015; Andrews et al., 2019), randomized and rounding-based approaches for multi-item OF (Jasin and Sinha, 2015; Ma, 2023), and Deep Controlled Learning (DCL) for sequential inventory-control problems (Temizöz et al., 2025). This enables a broader comparison than the common practice of benchmarking proposed policies primarily against simpler alternatives rather than established policies from the literature. Finally, Chapter 4 focuses on interpretable score-based policies, complementing the policy families studied in Chapter 3 by providing a representation closer to existing business logic. This aligns with prior work on compact, interpretable operational rules (Ferreira et al., 2022) and with recent data-driven fulfillment research that combines operational data, predictive models, and optimization (Ye et al., 2026).

The relationship between Chapters 3 and 4 is complementary. The framework in Chapter 4 can be extended to include objectives such as the vendor-satisfaction formulation introduced in Chapter 3. Conversely, the practical requirements emphasized in Chapter 4, such as data availability, interpretability, scalable instance generation, and compatibility with existing business logic, provide useful criteria for translating richer marketplace models from Chapter 3 into deployable allocation policies.

The thesis's contributions can therefore be summarized as a progression from understanding what OAOR is, to modeling what makes marketplace order allocation different, to building a practical DSF for evaluating and improving deployable allocation policies. This

progression is summarized in Figure 1.2, which also highlights the complementary relationship between the marketplace modeling contribution of Chapter 3 and the operational decision-support contribution of Chapter 4.

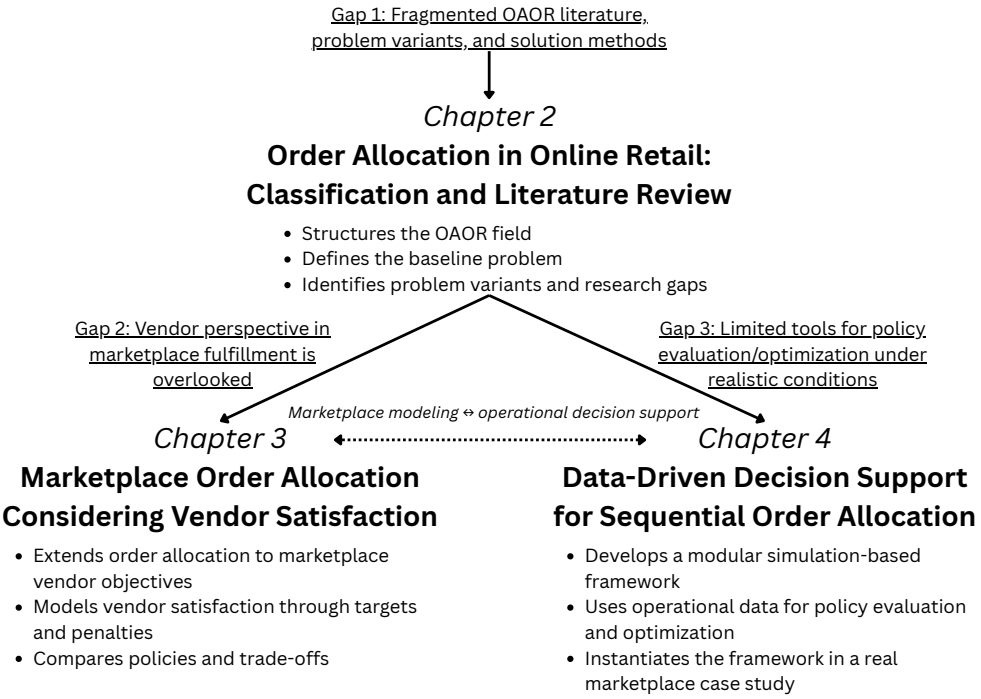


Figure 1.2: Thesis scope: from OAOR classification, to marketplace vendor-satisfaction modeling, to data-driven decision support for deployable allocation policies.

This thesis concludes with Chapter 5, entitled *General discussion, conclusions, and future research*, concludes the dissertation. It answers the research questions introduced in Section 1.2, synthesizes the findings of the three article-based chapters, and summarizes the main conceptual, modeling, methodological, and practical contributions of the thesis. The chapter also discusses managerial implications for retailers and marketplace operators. It concludes with a future research agenda.

Bibliography

Jason Acimovic and Stephen C. Graves. Making better fulfillment decisions on the fly in an online retail environment. *Manufacturing & Service Operations Management*, 17(1): 34–51, 2015. doi: 10.1287/MSOM.2014.0505.

Niels A.H. Agatz, Moritz Fleischmann, and Jo A.E.E. van Nunen. E-fulfillment and multi-channel distribution - a review. *European Journal of Operational Research*, 187(2): 339–356, 2008. doi: 10.1016/j.ejor.2007.04.024.

John M. Andrews, Vivek F. Farias, Aryan I. Khojandi, and Chad M. Yan. Primal–dual algorithms for order fulfillment at Urban Outfitters, Inc. *INFORMS Journal on Applied Analytics*, 49(5):355–370, 2019. doi: 10.1287/INTE.2019.1013.

Ethan Cramer-Flood. Worldwide retail ecommerce forecast 2024: Steadiness and stability replaces high highs and low lows. Technical report, EMARKETER, 2024. URL https://www.emarketer.com/content/worldwide-retail-ecommerce-forecast-2024. Accessed: 2026-05-25.

Cristiane Ferreira, Gonçalo Figueira, and Pedro Amorim. Effective and interpretable dispatching rules for dynamic job shops via guided empirical learning. *Omega*, 111:102643, 2022. doi: 10.1016/j.omega.2022.102643.

Alexander Hübner, Andreas Holzapfel, and Heinrich Kuhn. Operations management in multi-channel retailing: an exploratory study. *Operations Management Research*, 8(3-4):84–100, 2015. doi: 10.1007/S12063-015-0101-9.

Stefanus Jasin and Amitabh Sinha. An LP-based correlated rounding scheme for multi-item ecommerce order fulfillment. *Operations Research*, 63(6):1336–1351, 2015. doi: 10.1287/OPRE.2015.1441.

Sara Lebow. Ecommerce to account for more than 20% of worldwide retail sales despite slowdown, 2025. URL https://www.emarketer.com/content/ecommerce-account-more-than-20--of-worldwide-retail-sales-despite-slowdown. Accessed: 2026-05-25.

Will Ma. Order-optimal correlated rounding for fulfilling multi-item e-commerce orders. *Manufacturing & Service Operations Management*, 25(4):1324–1337, 2023. doi: 10.1287/msom.2023.1219.

Marco Melacini, Sara Perotti, Monica Rasini, and Elena Tappia. E-fulfilment and distribution in omni-channel retailing: a systematic literature review. *International Journal of Physical Distribution and Logistics Management*, 48(4):391–414, 2018. doi: 10.1108/IJPDLM-02-2017-0101.

Quan Yao Peh. Digital disruptors: The global online marketplace landscape in 2025, 2025. URL https://www.euromonitor.com/article/digital-disruptors-the-global-online-marketplace-landscape-in-2025. Accessed: 2026-05-25.

Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2219–2228. Association for Computing Machinery, 2018. doi: 10.1145/3219819.3220088.

Tarkan Temizöz, Christina Imdahl, Remco Dijkman, Douniel Lamghari-Idrissi, and Willem van Jaarsveld. Deep controlled learning for inventory control. *European Journal of Operational Research*, 324(1):104–117, 2025. doi: 10.1016/j.ejor.2025.01.026.

Ping Josephine Xu, Russell Allgor, and Stephen C. Graves. Benefits of reevaluating real-time order fulfillment decisions. *Manufacturing & Service Operations Management*, 11 (2):340–355, 2009. doi: 10.1287/MSOM.1080.0222.

Tinghan Ye, Sikai Cheng, Amira Hijazi, and Pascal Van Hentenryck. Contextual stochastic optimization for omnichannel multicourier order fulfillment under delivery time uncertainty. *Manufacturing & Service Operations Management*, 28(4):1068–1090, 2026. doi: 10.1287/MSOM.2024.1328.

CHAPTER 2

Order Allocation in Online Retail: Classification and Review

Order Allocation in Online Retail: Classification and Literature Review

Sérgio Vasconcelos* · **Gonçalo Figueira*** · **Bernardo Almada-Lobo***

Published in *European Journal of Operational Research*, 333(1), 1–18, 2026.

Abstract Online retail is transforming the way distribution networks are managed. One prominent change is that retailers can now use their full network to fulfil orders. This process involves allocating orders to fulfilment nodes and, depending on the setting, can include other operational decisions, such as order consolidation, shipping mode selection, and product substitution. This order allocation problem (OAOR) has garnered considerable attention in recent years. However, there is no comprehensive view of what has been done in the literature, nor a consistent terminology across papers, which makes it hard to position existing work and identify research gaps. To address these concerns, we conduct a systematic literature review, where we find over 60 articles contributing to the OAOR literature. From this review, we formulate the baseline problem, consider multiple extensions, and identify key problem characteristics. Additionally, we analyse and categorize the solution methods found based on the optimization mechanism, policy class, and incorporation of future information and learning. Our review points to several avenues for future research, both in problems and in solution methods.

Keywords E-commerce · Online Order Allocation · Fulfillment Optimization · Sequential Decision Making

2.1. Introduction

Online retail, also referred to as retail e-commerce, accounted recently for around 20% of global retail sales (Coppola, 2023). This share is projected to increase to around 24% by 2026, up to more than $8 trillion. E-commerce is highly relevant in several business sectors, including electronics, fashion, furniture, toys, hobby & DIY, beauty, health, personal & household care, and food and beverages. Moreover, online retail does not appear in

*INESC TEC, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal

complete isolation from traditional retail. Under a fulfillment model known as bricks-and-clicks, traditional retailers add an online channel to their portfolio, or online retailers open brick-and-mortar stores or collaborate with traditional retailers (Hübner et al., 2015). Initially, companies ran separate supply chains for online and traditional businesses, leading to fragmentation that negatively impacted customer experience and operational profitability. Joining both supply chains has resulted in omnichannel retailing, where the distinction between traditional and online retail is blurred (Melacini et al., 2018).

From a planning perspective, the fulfillment process is a key factor that distinguishes online retail from traditional retail. In traditional retail, setting up store inventory is the last stage of the delivery process since customers physically visit the stores to purchase items. In contrast, in online retail, the retailers move the items from their facilities to the customer's chosen delivery location. This allows the retailer to choose the most appropriate set of inventory-holding facilities (e.g., stores, warehouses) to fulfill each online order.

Therefore, the advent of online retail brought forth a new operational decision that opened up a new research stream. Since the (to the best of our knowledge) first attempt to address this issue by Xu et al. (2009), these efforts have resulted in significant growth and diversification of literature, encompassing different problem settings and solution methods. Researchers have expanded the scope of this problem to multiple e-fulfillment strategies, including pure online, multi-channel, and omnichannel strategies, and considered multiple problem dimensions, for example, optimizing for different allocation frequencies and integrating it with other operational decisions, such as dynamic pricing and delivery routing, utilizing a diverse set of solution strategies.

In practice, online orders arrive sequentially and can vary in size, ranging from orders that request only a single unit of a single item to orders that request multiple units of multiple items (Xu et al. (2009), Acimovic and Graves (2015), Jasin and Sinha (2015)). They are also characterized by a promised delivery date (e.g., next day, next week) and location (e.g., customer-provided address or a pick-up facility). The key decision in this problem, which we term OAOR, is allocating each unit of each item within an order to a unit of the same item in a fulfillment node (i.e., a facility holding inventory of that item) which can comply with those delivery constraints.

In the literature, the OAOR has been referred to by various names, such as the OF problem (Xu et al. (2009), Acimovic and Graves (2015)), including specific variants like the e-tail OF problem (Jasin and Sinha, 2015) or the omnichannel OF problem (Andrews et al., 2019), and the fulfillment-optimization problem (Acimovic and Farias, 2019). In our terminology, we intentionally avoid using the word *fulfillment* to clearly distinguish the problem of allocating online orders to fulfillment nodes from other e-fulfillment-related problems, such as delivery service design, pricing, forecasting, transportation planning, distribution network design, warehouse design, and inventory and capacity management (Agatz et al., 2008). It is important to note that while order allocation is distinguishable from the aforementioned problems, it is not independent. For one, it is influenced by tactical and strategic e-fulfillment decisions, e.g., distribution network design. Furthermore, it can be optimized alongside some operational decisions that influence inventory management, demand management, and the fulfillment process, extending the OAOR in different ways.

We note that the concept of *allocation* in retail is associated with multiple problems that are not the OAOR, such as order allocation to suppliers (Kawtummachai and Van Hop, 2005) and inventory allocation from distribution centers to stores (Federgruen and Zipkin, 1984). In the OAOR, the recipient of the allocation is the end customer, who has specificities that may extend the problem in various ways, e.g., considering pricing. Moreover, decisions are made more frequently (often on an order-by-order basis, rather than in replenishment cycles), thus requiring fast policies for this sequential decision-making process. Finally, in supplier selection, the suppliers' inventory is not known, which is a fundamental aspect of OAOR, to assess the future impact of allocation decisions.

The OAOR manifests in various contexts, each presenting unique challenges and considerations. For instance, in the realm of online marketplaces featuring products from multiple vendors, the dynamics of customer-vendor interactions vary. Some platforms empower customers to choose their preferred vendor, while others delegate this decision to the marketplace. In the latter scenario, the marketplace faces the task of determining a price before making order allocation decisions at checkout. This interdependence between pricing and order allocation underscores the importance of optimizing both decisions together. Within the same online marketplace context, vendors often sell their products not only on the platform but also through other channels such as their own stores or different marketplaces. Shared inventory between the online marketplace and external channels adds complexity, as inventory movements (replenishments and external sales) create a dynamic inventory state. This dynamic nature is further compounded by the conflicting optimization objectives of the primary stakeholders: the marketplace, vendors, and customers, each pursuing distinct goals during fulfillment.

Expanding the scope from pure online retail to traditional retailers adopting an omnichannel approach introduces further nuances. The execution of omnichannel strategies may involve exclusive online demand fulfillment from warehouses, stores, or a combination of both (see Taylor et al. (2019) for a review of omnichannel fulfillment strategies). Regardless of the strategy chosen, effective management and prioritization of demand allocation between online and traditional channels become crucial due to shared inventory. In omnichannel retail scenarios, where customers have orders delivered to a specific address, the OAOR process needs to be aligned with order delivery decisions, considering factors such as the delivery vehicles and time-window preferences specified by customers.

Effective online order allocation often depends on innovative solutions. In specific contexts, e.g., online grocery retail, customers are presented with substitutions of out-of-stock items for similar in-stock items. However, replacing such items may trigger a negative customer reaction, highlighting the delicate trade-off between lost sales and customer dissatisfaction. Effective stock management is paramount in industries with perishable products, such as food retail. The OAOR process plays a crucial role in optimizing order allocation, thereby aiding in the proper handling of perishable goods to mitigate the total shrinkage of the value chain. Errors and delays are often common in the delivery of online orders due to, e.g., picking and/or distribution errors or congestion. Implementing OAOR approaches with visibility over fulfillment nodes' characteristics that lead to these issues emerges as a valuable tool to mitigate the aforementioned challenges, contributing to a more efficient and streamlined fulfillment process. These examples illustrate the need to adapt the OAOR

to handle different context requirements correctly.

While there have been several contributions to the literature on the OAOR, addressing some of the aforementioned contexts and challenges, there is a lack of systematic exploration, description, and classification of different OAOR settings. In Acimovic and Farias (2019), the authors provide a tutorial on the OAOR where they identify four mechanisms to optimize the problem, which can be seen as different ways to formulate and solve the OAOR. The authors also categorize works based on the use of explicit forecasts, consideration of multi-item orders, and joint optimization with other decisions, such as pricing and web page display. However, their tutorial does not systematically capture all works related to the OAOR, overlooks important problem characteristics, and mostly focuses on only two solution approaches. Another review that classifies OAOR problems was conducted by Hübner et al. (2022), where the authors look into several problems in the context of omnichannel retail, including the *assignment of customer orders*. This also includes the assignment of orders in the context of strategic and tactical problems, such as facility location, where order allocation is just an auxiliary decision. The review classifies a wide range of works according to broad dimensions, while being limited to the omnichannel retail setting. Other reviews have also addressed OAOR in limited contexts. Qi et al. (2020) consider retail operations in general, but focus on data-driven methods, and do not classify the problems. Wang et al. (2021) consider the OAOR in the context of flexibility models, and focus on those whose performance measures are lost sales.

We believe that the reason for the lack of comprehensive reviews on the OAOR is twofold. First, the literature is relatively new, with most works appearing in the last five years. Second, research on the OAOR is scattered across various domains, making it difficult to capture and classify different works.

In light of the discussion above, our work makes the following contributions:

1. Systematically gather and review the literature that models and solve the OAOR and its extensions;

2. Create the first classification framework for the OAOR and its extensions, as well as the solution methods;

3. Identify research trends and opportunities for future research.

The remainder of the paper is organized as follows. Section 2.2 discusses the methodology used to collect relevant OAOR works. In Section 2.3, we delineate the primary dimensions of the main problem. The baseline problem and multiple extensions are introduced, while highlighting problem dimensions. Finally, we proceed to classify and review the literature identified in the preceding section in light of the dimensions identified. In Section 2.4, we analyze the fulfillment policies proposed in the literature. Section 2.5 highlights current research trends and opportunities for future research. Finally, Section 2.6, summarizes and concludes the paper.

2.2. Article collection procedure

Research on the OAOR is relatively new and dispersed across various domains. We conduct a systematic literature review, which is currently lacking, to identify contributions to this problem.

We start by defining the scope of the search according to the following criteria:

1. The article must propose a method to improve the decision of allocating orders to fulfillment nodes in the context of online retail or a multi-channel variant that includes an online channel, such as omnichannel retailing.

2. We consider articles that optimize the allocation of realized demand at the operational level. With this criterion, we exclude articles that propose models targeting strategic and/or tactical supply chain management problems (e.g., facility location, e.g., Aksen and Altinkemer (2008)), which use the allocation of estimated demand, not of actual orders, as an auxiliary decision.

3. The order allocation model proposed by the article needs to ensure that the chosen fulfillment node has the necessary inventory to fulfill each allocated order, to ensure that the allocation decisions are valid, that is, that the demand is correctly matched to supply. Recent works by Baron et al. (2024) and Ekren et al. (2025) assume that there is a probability that an item is available at fulfillment nodes but does not track the inventory state, therefore being excluded from this study. The inventory state is also not tracked in multi-agent fulfillment problems where multiple suppliers compete and/or collaborate to fulfill online orders, which resort to a function to compute the fitness between the order's requirements and the agent's bids. Papers that use this function, which does not account for the inventory state (e.g., Bhargava et al. (2016)), have been excluded.

4. We consider literature from two sources: (a) papers published in journals and conference proceedings indexed by Scopus or Web of Science, with no limit on publication date, and (b) papers published in repositories (e.g., SSRN and ArXiv), or published in journals and conference proceedings not indexed by Scopus or Web of Science, limited to publications from 2020 or later. Papers in (b) are queried through Google Scholar. We consider (a) our main source of literature, while (b) is used to gather recent literature that may have relevant contributions. A date limit is set for (b) since we expect significant works prior to 2020 to be published in a Scopus or Web-of-Science-indexed journal or conference.

5. The article must be written in English.

6. The search was limited to articles published before April 11th, 2024.

The above-mentioned criteria were applied in the article collection procedure, summarized in Figure 2.1. The procedure consists of three main stages: *identification*, where a set of potential articles was gathered, and *screening* and *eligibility*, where articles were analyzed and filtered according to the aforementioned criteria.

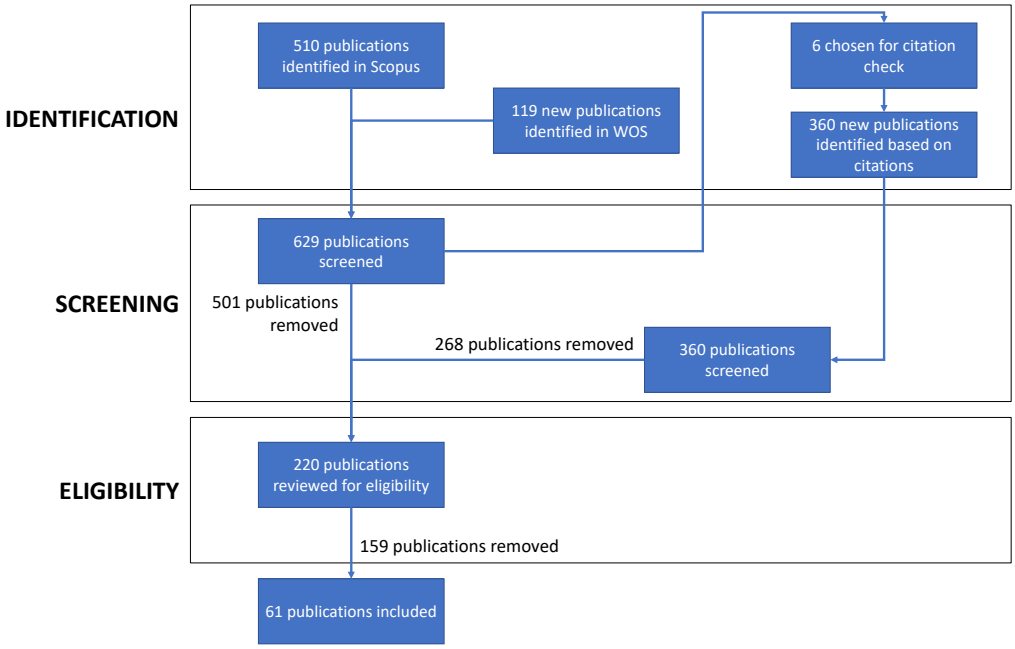


Figure 2.1: Article collection procedure.

Identification This stage occurred in two steps. The Scopus and Web Of Science databases were queried in the first step. Since the authors from different communities use different terms to describe the problem and its context, a set of exhaustive keywords was used in the query:

> *(((online OR omnichannel OR omni-channel OR multichannel OR multi-channel) AND retail*) OR etail OR e-tail) AND order* AND (fulfil* OR allocat* OR assign* OR rout*)*

With *TITLE-ABS-KEY* added at the beginning of the query for the Scopus search, and *(Topic)* added at the end of the query for the Web Of Science search. The first part of the query captures the different terms related to online retail and its variants. The second part captures the problems that may contain order allocation as a significant decision. Considering both databases, a total of 629 articles were initially identified.

The second step involved selecting six prominent articles for a citation check on Scopus, Web of Science, and Google Scholar. The aim of such procedure was to identify works that may not have been caught by the aforementioned query but fit our selection criteria. The aim of doing a citation check on Google Scholar, which is limited to works from 2020 or later, is explained in criterion 4. Five articles, namely Mahar and Wright (2009), Xu et al. (2009), Acimovic and Graves (2015), Jasin and Sinha (2015), and Torabi et al. (2015), were selected because they were the earliest OAOR publications with at least 40 citations on Scopus. The sixth article (Abdulkader et al., 2018) was selected to include an article addressing a VRP variant within the scope of the OAOR, which uses specific

terminology and is not often cited in other papers. In the end, 360 new publications were found in the second step, totaling 989 identified articles.

Screening The title, abstract and keywords of the 989 articles were checked to evaluate their compliance with the scope of the search. 652 articles were removed based on criteria 1-4 (483 from the first identification step, 169 from the second identification step), while 117 were removed based on criteria 5 and 6 (18 from the first identification step, 99 from the second identification step). In total, 769 publications were removed, leaving a total of 220 for the final step.

Eligibility The content of the 220 remaining articles was evaluated. In total, 152 articles were removed based on criteria 1-4. Additionally, seven were removed due to the inability to access the full text after attempting to contact the authors (Koc et al. (2017), Mathirajan (2021), Zhu et al. (2017), Azadiamin and Masel (2021), Li et al. (2020), Baraiya and Kapoor (2021), Karaoğlu and Kara (2022)). A total of 61 articles resulted at the end and hence constitute the basis of our study.

2.3. Problem Characterization

The OAOR has garnered dozens of significant contributions since it was first addressed by Xu et al. (2009). However, different characteristics separate multiple literature strings, effectively leading to different contributions to different sub-problems. In this section, we start by characterizing the baseline OAOR with a formulation that is general enough to represent all problem variations where the only decision is order-line allocation to fulfillment nodes. Later, we enrich our analysis by introducing other decisions related to multiple problem extensions.

As we explore the OAOR, we will introduce dimensions related to its *Environment* and *Objectives*. *Environment* dimensions characterize constraints within the retailer's operational environment, which includes other decisions that might affect order allocation. *Objectives* dimensions characterize the goals the retailer wishes to optimize in the context of OAOR.

2.3.1 Baseline Problem

We define the baseline OAOR as the problem of allocating a unit of inventory at a fulfillment node to each order-line, i.e., each unit of item within an order. This baseline problem can thus be defined as follows (see notation in Table 2.1). Let t be a period of time, fixed or variable, during which a set of orders O_t arrive, and T be a number of periods to be considered. The problem consists of determining the amount of demand of item $i \in I$ in each order $o \in O_t$, $d_{o,i}$, to be allocated to fulfillment node $j \in J$, $x_{o,i,j}$, so as to minimize a given cost function, $fc_t(.)$, while ensuring non-negative inventories in all periods, $s_{i,j,t}$. The formulation is given by:

$$\text{Minimize:} \quad \sum_{t=1}^{T} fc_t(\boldsymbol{x}_t) \tag{2.1}$$

$$\text{Subject to:} \quad s_{i,j,t} = s_{i,j,t-1} + s^{ext}_{i,j,t} - \sum_{o \in O_t \cap O_j} x_{o,i,j} \forall i \in I, j \in J_i, t \in \{1,...,T\} \tag{2.2}$$

$$\sum_{j \in J_i \cap J_o} x_{o,i,j} = d_{o,i} \qquad \forall o \in O, i \in I \tag{2.3}$$

$$x_{o,i,j} \in \mathbb{N}_0 \qquad \forall o \in O, i \in I, j \in J \tag{2.4}$$

$$s_{i,j,t} \in \mathbb{N}_0 \qquad \forall i \in I, j \in J, t \in \{1,...,T\} \tag{2.5}$$

Table 2.1: Notation for Indexes, Sets, Variables, and Parameters

Indexes/Sets	
$0 \leq t \leq T$	Period
$i \in I$	Item
$j \in J$	Fulfillment Node
$o \in O$	Order
O_t	Set of orders that arrive in period t
t_o	Period of arrival of order o
J_i	Set of fulfillment nodes that can fulfill item i
J_o	Set of fulfillment nodes that can fulfill order o
O_j	Set of orders that can be fulfilled by fulfillment node j
Variables	
$x_{o,i,j}$	Amount of demand of item i in order o allocated to fulfillment node j; $\boldsymbol{x}_t = (x_{o,i,j})_{i \in I, j \in J, o \in O_t}$
$s_{i,j,t}$	Inventory of item i in fulfillment node j at the end of period t (for $t = 0$, it represents the initial inventory); $\boldsymbol{s}_t = (s_{i,j,t})_{i \in I, j \in J}$
Parameters	
$fc_t(\boldsymbol{x}_t)$	Function that computes fulfillment costs of the demand allocated in period t
$d_{o,i}$	Demand of item i in order o
$s^{ext}_{i,j,t}$	External variation of inventory of item i in fulfillment node j during period t; $\boldsymbol{s}^{ext}_t = (s_{i,j,t})_{i \in I, j \in J}$

Equation (2.1) represents the objective function, which is to minimize total fulfillment costs. Constraints (2.2) ensure the inventory is correctly accounted and carried between periods. Constraints (2.3) ensure that all demand is fulfilled. Constraints (2.4) and (2.5) define the variable's domain.

The above model represents a static view of the problem. However, in practice, information is revealed and the problem is solved dynamically, as follows:

For $t = 1$ to $t = T$:

1. If $t = 1$, information on the initial inventory state, $\boldsymbol{s}_{t=0}$, is revealed;
2. [Make decisions that occur prior this period's order(s) arrival (e.g. pricing, replenishment)];
3. Observe external variations of inventory, $\boldsymbol{s}^{ext}_t$;
4. Orders that arrive in period t, O_t, are revealed;

5. Irrevocable order allocation decisions for all orders in O_t, $\boldsymbol{x}_t$, are made. [Make other decisions that occur simultaneously with order allocation (e.g. cartonization, routing)];
6. Inventories at the end of period t, $\boldsymbol{s}_t$, are calculated;
7. Fulfillment costs for the demand allocated are computed following $fc_t(\boldsymbol{x}_t)$. [Other optimization objectives for period t are computed].

Note that sentences inside square brackets are only relevant for problem extensions, which we discuss in Subsection 2.3.2.

The baseline model makes the following assumptions:

A. There is always enough inventory to fulfill all demand, such that $\sum_{o \in O_t \cap O_j} x_{o,i,j} \leq s_{i,j,t-1} + s^{ext}_{i,j,t}, \quad \forall i \in I,\ j \in J_i,\ t \in \{1, \ldots, T\}$. This ensures that a feasible solution always exists;

B. Order allocation decisions cannot be delayed.

2.3.1.1 Environment dimensions

Order allocation timing The foremost environmental dimension of the OAOR is the order allocation timing. There are two main timings for allocation decision-making: in batches and order-by-order (or, equivalently, immediately upon arrival).

- *Order-by-Order (ObO)* – A single order is allocated at a time, thus $|O_t| = 1$. The length of period t is variable, depending on the time elapsed between subsequent order arrivals.
- *Batch* – Multiple orders are allocated at a time, thus $|O_t| > 1$. Period t generally has a fixed length (e.g., one day). Within *batch* allocation, we distinguish two particular settings:
  - *Single-Period Batch (SP Batch)* – When $T = 1$, such that there is only one order batch to allocate;
  - *Multi-Period Batch (MP Batch)* – When $T > 1$, such that there are multiple order batches to allocate.

Whether to formulate and solve an OAOR as a ObO, SP Batch or MP Batch problem depends on multiple practical considerations:

- Operational constraints – The operational reality of the retailer often aligns best with a specific order allocation timing. For example:
  - Delivery Deadlines – the shorter the delivery deadlines, the more suited a Batch timing with small batch sizes or an ObO timing is;
  - Retailer-Customer Interaction – it may dictate an ObO timing if information about the fulfillment node is provided at checkout;

  - Fulfillment Node Picking Schedules – Order allocation can be set to occur immediately before picking instructions are sent to fulfillment nodes, allocating batches of orders that arrive between subsequent picking waves.

- Approximation to a static problem – Addressing the OAOR often involves simplifying its dynamic nature into a static approximation, enabling the use of simpler methods like mixed-integer programming and heuristics. A common strategy consists of converting a MP Batch problem into multiple SP Batch problems, one for each order batch. However, this simplification neglects the impact of future orders on current allocations, leading to sub-optimal outcomes, as highlighted by Acimovic and Graves (2015). While static approximations may be attractive in scenarios where sub-optimality is less pronounced or absent, this work does not aim to identify such situations. Still, we underscore the importance of considering this factor when modeling and solving the OAOR.

Number of items and units The terminology regarding order composition and size can be ambiguous. According to Xu et al. (2009), a "single order" refers to a customer order that consists of exactly one unit of one SKU, while a "multiorder" is a customer order that includes more than one unit, either of one SKU or multiple SKUs. Later, Andrews et al. (2019) used the term "multiproduct order" interchangeably with "multiorder". However, other authors like Acimovic and Graves (2015) and Jasin and Sinha (2015) introduced the terms "single-item order" and "multi-item order" to describe orders with one or multiple products, respectively. The terms "item", "product", and "SKU" are used interchangeably in the literature.

In an attempt to simplify nomenclature, we define the following dimensions characterizing an order's composition and size:

- Number of items – The number of items within the order. We consider two situations:
  - *Single-Item (1P)* – orders can only contain a single item, such that for $o \in O$: $d_{oi} > 0 \rightarrow d_{oi'} = 0 \; \forall i' \in I, i' \neq i$;
  - *Multi-Item (mP)* – there is no limit to the number of items;

- Number of units per item – The number of units demanded for each item. We consider two situations:
  - *Single-Unit (1U)* – each order can only demand a unit of each item, such that $d_{o,i} \leq 1 \; \forall o \in O, i \in I$;
  - *Multi-Unit (mU)* – there is no limit to the number of units ordered for each item.

While in practice most retailers sell multiple items, they may decide to optimize order allocation separately for each item, converting a mP problem to multiple 1P problems. However, it should be noted that such approximations can also result in sub-optimality.

Again, while we do not identify the specific conditions when this occurs, we call to its importance in practice.

It is important to clarify why we consider the number of units per item demanded in each order an important dimension, as it may seem a bit nit-picking. We make this distinction because some solution methods compute marginal values and require the assumption that demand is unitary, e.g. Jasin and Sinha (2015) and Andrews et al. (2019).

Fulfillment node network flexibility The fulfillment node network can be primarily characterized by its Flexibility, which can be related to the *item* or the *order*. We distinguish two main flexibility types:

- Full – every fulfillment node can fulfill demand from any item ($J_i = J, \forall i \in I$) or demand from any order, such that $J_o = J, \forall o \in O$;
- Limited – certain items are not available in some fulfillment nodes ($J_i \subset J, \exists i \in I$), or nodes are not allowed to fulfill certain orders, due to constraints on the delivery region or facility level ($J_o \subset J, \exists o \in O$). Some types of networks with limited flexibility are long-chain (LC) networks (Asadpour et al., 2019), generalized chaining gaps (GCG) networks (Xu et al., 2020), and dual fulfillment node network, when only two fulfillment nodes can fulfill an order (in practice, retailers often allow only orders to be allocated to one of the two closest fulfillment nodes, as in Wei et al. (2020), or to either the customer's local or regional fulfillment node, as in Zhao et al. (2025)).

Generally, the more flexible, the more fulfillment choices, making the problem harder to solve. Hence, in practice, retailers impose conditions on the number of fulfillment nodes that can fulfill an order and/or an item.

Fulfillment node types Fulfillment nodes, in practice, exhibit different characteristics that enable them to be utilized differently in fulfilling demand in online and omnichannel retail contexts. Taylor et al. (2019) identifies multiple fulfillment strategies in omnichannel retail, namely, Buy Online Pick Up In Store (BOPS), Ship To Store / Buy Online Ship To Store (STS), Buy Online Ship From Store (BOSS), Omnichannel Distribution Center (ODC), Omnichannel Drop-Shipping (ODS) and Buy Online Return In Store (BORIS). ODS and BORIS require utilizing drop-shipping and managing order returns, respectively, which we treat as extensions to the baseline OAOR considered in Subsection 2.3.2. As such, we focus on the remaining strategies: BOPS, STS, BOSS, and ODC.

An aspect that distinguishes these strategies is where the customer collects the order: at a facility (e.g., a store or locker), often called *click-and-collect* modes, in BOPS and STS strategies, or at a customer-specified address (e.g., home), in BOSS and ODC strategies. When modelling the order allocation problem, there is no substantial difference between the two. This is because the allocation of order demand to fulfillment nodes translates into fulfillment costs $fc_t(\boldsymbol{x}_t)$ that are a function of the origin (fulfillment node location)

and destination, where the destination can be interpreted both as a customer-given address or a pick-up facility. While we recognize that most papers reviewed state that fulfillment/delivery is directed to a "customer", which implies that it is a customer-specified address, the models are also applicable to click-and-collect strategies if the locations are interpreted as facilities.

Another relevant aspect is the types of facilities that are used as fulfillment nodes by different strategies, namely stores (in BOPS and BOSS) and Distribution Center (DC) (in STS and ODC). The difference between these facilities lies in the roles that their inventory has:

- Fulfill retailer-allocated demand – the fulfillment of orders that the retailer centrally allocates to a fulfillment node that adheres to the network's flexibility constraints;
- Fulfill customer-allocated demand – the fulfillment of orders at the customer's initiative, typically through physical presence (walk-in customers) or explicit choice (BOPS*), such that, for those orders, $|J_o| = 1$;
- Replenish other facilities – for a fulfillment node j replenishing item i in another facility in period t, $s^{ext}_{i,j,t} < 0$.

In the context of the OAOR, inventory at fulfillment nodes always fulfills the role of fulfilling retailer-allocated demand. The combination of this with other purposes leads to different types of fulfillment nodes:

- Online DC – fulfillment nodes whose inventory is used exclusively to fulfill retailer-allocated demand;
- Store (S) – fulfillment nodes whose inventory is used to fulfill retailer-allocated demand and customer-allocated demand;
- ODC – fulfillment nodes whose inventory is used to fulfill retailer-allocated demand and to replenish other facilities.

The exclusive use of Online DC characterize pure online retail settings, whereas the use of Stores and ODC are indicative of omnichannel retail contexts. The literature surveyed showed that different types of facilities are often included in the same OAOR model, as evident in Table 2.3.

External inventory movements External inventory movements, $\boldsymbol{s}^{ext}_t$, are changes to the inventory induced from an external source, i.e., that are not a result of decisions taken within the model. These could be, e.g., inventory replenishments, transshipments, or even represent inventory perishability. We distinguish works according to their external inventory movements as follows:

*Given that the customer requests pick up at a specific store, and the inventory used to fulfill it comes from the store's on-hand supply, the allocation decision does not exist.

- No external inventory movements – when the initial inventory is used exclusively (there are no external replenishments) and there are no other sources of inventory depletion rather than order allocation decisions $\boldsymbol{x}_t$. In these models, we can simplify constraints (2.2) to $s_{i,j,t} = s_{i,j,t-1} - \sum_{o \in O_t} x_{o,i,j}$.

- With external inventory movements – when external replenishments and/or depletions are included in the model.

None of the literature included in our review considers external inventory movements.[†] Therefore, for the remainder of this work, we consider a simplified baseline problem with the aforementioned simplified version of the constraints (2.2).

2.3.1.2 Objectives dimensions

There are two main approaches to model objectives in OAOR contexts:

- Single objective function – the most prevailing approach is to use a function that assigns numeric rewards or costs to order allocation decisions. In practice, as pointed out by Andrews et al. (2019), this function can be set up to reflect one or multiple objectives, such as maximizing profit, minimizing shipping cost, minimizing the delivery lead time and/or minimizing picking errors. However, as the number of objectives increases, developing a single reward function that accurately captures each objective as intended becomes more challenging.

- Multiple objective functions – Use multiple functions that reflect specific fulfillment objectives, generally when objectives are measured in different units (e.g. shipping costs, in dollars, and delivery lead time, in hours).

In our baseline problem formulation, we consider a single objective function reflecting a single objective, the minimization of *Fulfillment Costs* ($fc_t(\boldsymbol{x}_t)$). This represents the cost of preparation plus direct delivery (from the fulfillment node to the delivery address). Fulfillment cost components can be "bundled" into a single parameter or separated according to their specific activities. This separation becomes important when fulfillment processes differ from order to order. For example, when customers pick up orders in-store, delivery costs are not incurred if store inventory is used, contrary to orders where home delivery is requested. These costs usually take the form of a fixed and/or variable component. For example, Jasin and Sinha (2015) defines a fulfillment cost function where the fixed component represents a per-shipment cost, while the variable component varies based on the number of units in the shipment. Fulfillment cost functions with fixed cost components are subject to consolidation gains; in other words, allocating multiple order lines to the same fulfillment node or sharing other fulfillment resources, such as delivery vehicles, leads to cost savings since the cost of adding more order lines to the same resource marginally decreases.

[†]While we found literature that includes inventory replenishment and transshipment decisions, these are always taken as decisions within the model.

2.3.2 Problem Extensions

We now introduce several extensions to the base model by jointly modeling order allocation with other operational decisions, which we group by decision types:

- **2.3.2.1 Unfulfillment/Drop-Shipping and Allocation Delay** – Decisions that extend the baseline model by relaxing assumptions A and B. These decisions occur simultaneously with order allocation (Step 4 in Subsection 2.3.1).
- **2.3.2.2 Inventory Management (IM)** – Decisions that, jointly with order allocation, alter the inventory state, $\boldsymbol{s}_t$. These decisions generally occur prior to order allocation (Step 2 in Subsection 2.3.1).
- **2.3.2.3 Demand Management (DM)** – Decisions that alter customer demand, $d_{o,i}$. These decisions occur prior to order allocation (Step 2 in Subsection 2.3.1), except Order Acceptance and Product Substitution decisions, which take place immediately upon each order arrival.
- **2.3.2.4 Fulfillment Process Management (FPM)** – Decisions, including order preparation and delivery tasks, that, jointly with order allocation, alter fulfillment costs, $fc_t(\boldsymbol{x}_t)$. These decisions occur simultaneously with order allocation (Step 4 in Subsection 2.3.1).

As we introduce each decision, which corresponds to a new *environment* dimension, we also introduce new *objective* dimensions that are often associated with those decisions and briefly explain possible ways to extend the base model introduced in Subsection 2.3.1. We summarize the information on the new decisions and objectives introduced by the Problem Extensions in Table 2.2. Note that, within the "Decision / Environment Dimension", the type of the decisions corresponds to the grouping of decisions we previously discussed, whereas within the "Objective Dimensions", we introduce the following types: (a) Loss of Customer Goodwill, which is related to customer dissatisfaction, (b) Revenue and Profit, which are revenues and profits related to some DM decisions, (c) Reverse-Logistic Cost that relates directly to order returns, (d) Inventory Cost, related to IM operations, and (e) Fulfillment Cost, initially introduced in 2.3.1.2 within the context of the Baseline Model. With the introduction of some decisions, these latter costs are not solely tied to order allocation but also encompass other decision-making aspects, becoming attributable to other decisions. Within the Fulfillment Cost type, we distinguish between costs related to preparation tasks, costs related to delivery tasks, and those that can be attributable to both. Finally, we note that some objectives do not have a specific decision related to them (denoted by "-"), but often appear in the context of other decisions of a certain type, as we later explain.

2.3.2.1 Unfulfillment/Drop-Shipping and Allocation Delay

Proactive/Reactive (P/R) Demand Unfulfillment/Drop-Shipping If we drop assumption A of the Base Problem (Subsection 2.3.1), there may be no feasible options to allocate

Table 2.2: Summary of Environment and Objective Dimensions based on Problem Extensions

Decision / Environment Dimensions			**Objective Dimensions**	
Type	***Acronym***	***Decision***	***Type***	***Objective***
Unfulfillment/Drop-Shipping and Allocation Delay	R	Reactive Unfulfillment/Drop-shipping	Loss of Customer Goodwill, if Unfulfillment Fulfillment Cost (Preparation + Transportation), if Drop-shipping	Lost Sales Cost, if Unfulfillment Drop-Shipping Cost, if Drop-Shipping
	P	Proactive Unfulfillment/Drop-shipping		
	ADL	Allocation Delay	Loss of Customer Goodwill	Delay Cost - Penalty only for delayed fulfillment - Per-period penalty
Inventory Management	I-I	Initial Inventory	Inventory Cost	Inventory Ordering and Allocation Costs
	I-RP	Inventory Replenishment		
	I-TR	Proactive Inventory Transshipment		Inventory Transshipment Cost
	-			Inventory Holding Cost
Demand Management	D-PR	Product Pricing	Revenue and Profit	Price/Framing revenue
	D-FR	Product Framing		
	D-CO	Channel Opening/Closure		Channel Profit (can include costs, e.g. channel commission, and revenues, e.g. channel selling price)
	D-OR	Order Rejection	Loss of Customer Goodwill	Lost Sales Cost
	D-SU	Product Substitution		Substitution Cost
Fulfillment Process Management	F-OP	Order Picking	Fulfillment Cost (Preparation)	Picking Cost
	F-CA	Order Cartonization		Cartonization Cost
	F-OC	Order Consolidation	Fulfillment Cost (Transportation)	Order Consolidation Cost
	F-OR	Order Routing		Order Routing Cost
	F-MS	Fulfillment Mode Selection	Fulfillment Cost (Preparation + Transportation)	Fulfillment Mode Cost
	F-TR	Reactive Inventory Transshipment		Inventory Transshipment Cost
	-		Reverse-Logistics Cost	Order Return Cost

an order line at the time of allocation. In such situations, an order line can be left *unfulfilled* and/or *drop-shipped* (i.e., fulfilled from an external supplier at a higher cost). These actions may be taken *reactively* (R), when there are no other fulfillment options available, or *proactively* (P), when there are fulfillment options available, but the retailer chooses to reserve inventory for more profitable orders in the future. Unfulfillment results in a considerable loss of customer goodwill; whereas drop-shipping incurs in higher fulfillment costs compared to normal fulfillment.

We found literature to either consider unfulfillment or, as an alternative, drop-shipping, and never both options simultaneously. Authors frequently allocate order lines to a fictitious fulfillment node (FF), representing either unfulfillment or drop-shipping, and the associated allocation cost to be a cost of loss of customer goodwill or cost of fulfillment through drop-shipping, respectively. The base model can be trivially extended to consider the case of a FF: (1) consider J to be the set of all real fulfillment nodes plus a fictitious fulfillment node, corresponding to j^f, (2) assume $s_{i,j^f,t} = \infty \; \forall i,t$, and (3) adapt $fc_t(\boldsymbol{x}_t)$ to reflect the high allocation costs to a FF (if only reactive allocations to the FF would occur, then an extra constraint would be required to only allocate to the FF if there is no feasible non-FF allocation).

Allocation Delay (ADL) If we drop assumption B of the Base Problem (Subsection 2.3.1), demand is allowed to be allocated in a subsequent period. In practice, this can occur, for example, if the order's delivery deadline is not compromised, and/or if a penalty is considered for allocations whose delivery deadlines are compromised, or simply a per-delayed-period time penalty if no delivery deadline is explicit.

Considering $x^d_{o,i,j,t}$ to be the amount of demand of item i in order o allocated to fulfillment node j in period $t > t_o$, $\boldsymbol{x}^d_t = (x^d_{o,i,j,t})_{i\in I, j\in J, o\in\bigcup_{t'<t} O_{t'}}$, and $dc_t(\boldsymbol{x}^d_t)$ the function that computes ADL costs in period t, then by (1) including the term $\sum_t dc_t(\boldsymbol{x}^d_t)$ in the objective function (2.1), (2) modifying constraints (2.2) to $s_{i,j,t} = s_{i,j,t-1} - \sum_{o\in O_t} x_{o,i,j} - \sum_{o\in\bigcup_{t'<t} O_{t'}} x^d_{o,i,j,t}$, and (3) modifying constraints (2.3) to $\sum_{j\in J_i\cap J_o} x_{o,i,j} + \sum_{t'>t_o, j\in J_i\cap J_o} x^d_{o,i,j,t'} = d_{o,i}$, we extend the base model to allow delayed allocation considering penalties for delayed allocation.

2.3.2.2 Inventory Management (IM)

Initial Inventory (I-I) Determine the total initial item inventory ordering amount and their allocation across fulfillment nodes.

Considering $s_{i,j,t=0}$ to be a variable representing the initial inventory instead of a parameter, $\boldsymbol{s}^0 = (s_{i,j,t})_{i\in I, j\in J, t=0}$, and $iic(\boldsymbol{s}^0)$ the function that computes I-I ordering and allocation costs, we extend the base model to consider I-I ordering and allocation decisions by adding $iic(\boldsymbol{s}^0)$ to the objective function (2.1).

Inventory Replenishment (I-RP) Determine the total item inventory ordering amount and their allocation across fulfillment nodes between order allocation periods, adding to existing inventory carried from the previous period.

Considering $s^+_{i,j,t}$ to be the amount of inventory of item i replenished at fulfillment node j in period t, $\boldsymbol{s}^+_t = (s^+_{i,j,t})_{i\in I, j\in J}$, and $irc_t(\boldsymbol{s}^+_t)$ the function that computes inventory ordering and allocation costs at period t, we can extend the baseline model to consider I-RP decisions by adding $irc_t(\boldsymbol{s}^+_t)$ to the objective function (2.1) and modifying constraints (2.2) to $s_{i,j,t} = s_{i,j,t-1} + s^+_{i,j,t} - \sum_{o\in O_t} x_{o,i,j}$.

Note that some works decouple inventory ordering from inventory allocation decisions. In Lim et al. (2020) and Jiu (2022), inventory ordering decisions are made before demand is revealed (Step 2 in Subsection 2.3.1), while inventory allocation decisions are made simultaneously with order allocation (Step 4).

Proactive Inventory Transshipment (I-TR) Decisions on proactively relocating inventory between fulfillment nodes. Two situations can occur: (1) the fulfillment nodes are in the same echelon, thus, the transshipment functions as an adjustment; (2) the fulfillment nodes are in different echelons (e.g., from DC to stores), which in practice functions as a replenishment. Within the literature surveyed in our review, no model includes (1), and only a single work admits (2) (Jiu, 2022). In the latter, some facilities include inventory that is thus pooled to serve the demand of flexible source orders and replenish other facilities. We consider reactive inventory transshipment to be transshipment that occurs as a reaction to customer orders, thus being part of the OF process. Contrarily, I-TR aims to rebalance the inventory network state for future OF.

Considering $s^{tr}_{i,j,j',t}$ to be the amount of inventory of item i transshipped from fulfillment node j to fulfillment node j' in period t, $\boldsymbol{s}^{tr}_t = (s^{tr}_{i,j,j',t})_{i\in I, j\in J, j'\in J\setminus j}$, and $itc_t(\boldsymbol{s}^{tr}_t)$ the function that computes inventory transshipment costs at period t, the base model can be extended to include proactive transshipment by adding $\sum_t itc_t(\boldsymbol{s}^{tr}_t)$ to the objective function (2.1) and modifying constraints (2.2) to $s_{i,j,t} = s_{i,j,t-1} - \sum_{j'\in J\setminus j} s^{tr}_{i,j,j',t} + \sum_{j'\in J\setminus j} s^{tr}_{i,j',j,t} - \sum_{o\in O_t} x_{o,i,j}$.

When IM decisions are considered, the objective of minimizing inventory holding costs is also often considered. However, this objective can also be included in the baseline problem since order allocation decisions effectively manage the inventory state on their own.

Assuming $ihc_t(\boldsymbol{s}_t)$ to be the function that computes inventory holding costs at period t, inventory holding costs can be considered in the baseline model by adding $\sum_t ihc_t(\boldsymbol{s}_t)$ to the objective function (2.1).

2.3.2.3 Demand Management (DM)

Product Pricing (D-PR) and Product Framing (D-FR) Determining the price (D-PR) and display option (D-FR), the latter of which may include the product price, for each product in the online channel. These decisions allow retailers to increase (decrease) desirability for products with better (worse) fulfillment conditions.

Considering $w \in W$ to be a display option (which could simply be a price), $y_{w,i,t}$ to take value of 1 if display option w is chosen for item i in period t, 0 otherwise, $\boldsymbol{y}_t = (y_{w,i,t})_{w\in W, i\in I}$, and $dpr_t(\boldsymbol{y}_t, \boldsymbol{x}_t)$ to be a function that computes the revenue associated with the display options chosen in period t, then extending the base model to include dynamic pricing/framing decisions can be done by considering an objective function that maximizes $\sum_t dpr_t(\boldsymbol{y}_t, \boldsymbol{x}_t)$

minus the fulfillment costs in (2.1). Constraints would remain the same, but $d_{o,i}$ is now function of the price/display option decisions $y_{w,i,t}$.

In some cases, the display option choice may limit the feasibility of order-line allocation options, requiring additional constraints in the model. In Bai et al. (2025), a display option represents a limit on the delivery time (e.g., 1-day, 2-day), which constrains the feasible fulfillment nodes to those respecting the delivery time.

Maximizing the revenue earned can also be considered in the baseline model, as items may have different prices if fulfilled through different fulfillment nodes (e.g., in marketplaces, if inherent to the choice of a fulfillment node is a vendor choice).

Channel Opening/Closure (D-CO) Determining the opening and closure of specific sales channels. This decision depends on the system's inventory state and the profitability of each channel.

The baseline problem can be extended to include D-CO decisions similar to the extension to include D-PR or D-FR. The differences would be that w would now represent a sales channel and $y_{w,t}$ to take value of 1 if channel w is open during period t, 0 otherwise, with the dependence on the item i dropped. $dpr_t(\boldsymbol{y}_t, \boldsymbol{x}_t)$ would now be a function that computes the profit associated with channels open and closed during period t, and $d_{o,i}$ would be a function of the channel decisions $y_{w,t}$.

Order Rejection (D-OR) Decision on rejecting customer orders immediately upon order arrival, in practice incurring a loss of customer goodwill. This decision is only applicable to Batch problems, since in ObO it would be equivalent to proactive demand unfulfillment.

Considering z_o to take value 1 if order o is accepted, 0 otherwise, $\boldsymbol{z}_t = (z_o)_{o \in O_t}$, and $drc_t(\boldsymbol{z}_t)$ to be a function that computes rejection costs at period t, then the base model can be extended to consider demand rejection by adding $\sum_t drc_t(\boldsymbol{z}_t)$ to the objective function (2.1) and modifying constraints (2.3) to $\sum_{j \in J_i \cap J_o} x_{o,i,j} = d_{o,i} \times z_o$.

Product Substitution (D-SU) Substituting an out-of-stock or low-availability product for a similar in-stock or highly available product.

Considering $h_{o,i,i'}$ to be the amount of demand of product i in order o replaced by product i', and $dsc_{o,i,i'}(h_{o,i,i'})$ a function that computes the cost of substituting item i with i' in order o, reflecting the customer's substitution preferences, the baseline model can be extended to include D-SU by adding the term $\sum_{o \in O} \sum_{i \in I} \sum_{i' \in I \setminus \{i\}} dsc_{o,i,i'}(h_{o,i,i'})$ to the objective function (2.1), modifying constraints (2.3) to $\sum_{j \in J_i \cap J_o} x_{o,i,j} = d_{o,i} + \sum_{i' \in I \setminus \{i\}} h_{o,i',i} - \sum_{i' \in I \setminus \{i\}} h_{o,i,i'}$, and adding constraints $\sum_{i' \in I \setminus \{i\}} h_{o,i,i'} \leq d_{o,i}$ for all $o \in O, i \in I$.

2.3.2.4 Fulfillment Process Management (FPM)

Order preparation decisions Includes decisions such as **Order Picking (F-OP)** (i.e., selecting and retrieving items) and **Order Cartonization (F-CA)** (i.e., determining the order packaging, including the number of packages, its size and type), which occur at the allocated fulfillment node.

For example, to extend the base model to include F-CA decisions, we could consider $c \in C$ to be the carton type, $sizeCarton_c$ to be the dimensional capacity of carton type c, $sizeItem_i$ to be the dimensional measure of item i, $processCarton_c$ to be the unitary processing requirement of carton type c, $processNode_{j,t}$ to be the processing capacity of fulfillment node j in period t, $k_{o,j,c}$ to be the number of cartons of type c used to fulfill order o from fulfillment node j, $\boldsymbol{k}_t = (k_{o,j,c})_{j\in J, c\in C, o\in O_t}$, and $dcc_t(\boldsymbol{k}_t)$ to be the cartoning costs in period t, and (1) add $\sum_t dcc_t(\boldsymbol{k}_t)$ to the objective function (2.1), (2) add constraints $\sum_c sizeCarton_c \times k_{o,j,c} \geq \sum_i sizeItem_i \times x_{o,i,j}\ \forall o, j$, and (3) add constraints $\sum_{c,o\in O_t} processCarton_c \times k_{o,j,c} \leq processNode_{j,t}\ \forall j, t$.

Order delivery decisions Includes all the activities that determine how the inventory physically moves from the fulfillment nodes to the order's delivery address, e.g., **Fulfillment Mode Selection (F-MS)** (e.g., selecting the delivery carrier), **Order Consolidation (F-OC)** (which facility to consolidate each order before its final delivery), **Order Routing (F-OR)** (the path taken by vehicles that collect orders from fulfillment nodes and delivers them to customers) and **Reactive Inventory Transshipment (F-TR)** (transshipment of inventory that is already assigned to an order).

These decisions are often associated with the sharing of order shipments with other deliveries. All works surveyed with F-OR aggregate multiple customer orders in delivery routes. Furthermore, multiple works that combine F-OR with Inventory Replenishment consider shipments that carry both deliveries and replenishment (Abdulkader et al. (2018), Bayliss et al. (2020), Martins et al. (2020b), Martins et al. (2020a), Martins et al. (2021) and Gao et al. (2022)). Akyüz et al. (2022) considers the aggregation of multiple customer orders up to a consolidation point, from which the final delivery is carried out individually per order. Torabi et al. (2015) considers the aggregation of shipments within transshipments.

For example, to extend the base model to include F-OC decisions, we could consider L to be the set of non-fulfillment nodes that, alongside fulfillment nodes, can be used for consolidation; $c_{o,j}$ to be a binary variable taking value 1 if node $j \in L \cup J$ is used as a consolidation node for order o, and 0 otherwise; $\boldsymbol{c}_t = (c_{o,j})_{o\in O_t, j\in L\cup J}$; and fulfillment costs to depend on both order allocation and consolidation decisions, such that $fc_t(\boldsymbol{x}_t, \boldsymbol{c}_t)$. Then, the base model can be extended by (1) replacing the objective function (2.1) by $\sum_t fc_t(\boldsymbol{x}_t, \boldsymbol{c}_t)$ and (2) adding constraints $\sum_{j\in L\cup J} c_{o,j} = 1, \forall o$.

Two optimization objectives that, although not necessarily tied to any FPM decision, are often associated with them is minimizing the costs of delayed delivery[‡] and/or order return. Indeed, in practice, due to differences in fulfillment processes, e.g. congestion at the picking or delivery stage, an allocation to different fulfillment nodes could result in different rates of delayed order delivery and/or order return, making it an important consideration. We note that none of the literature included in our study considers order returns.

[‡]This is distinct from the allocation delay previously mentioned. Whereas here we are considering a delivery delay, that is, the order being received by the customer after its delivery deadline, in ADL we are considering delaying the allocation decision in itself, which does not imply that the order will arrive at the customer after its delivery deadline.

Considering $rc_{i,j,t}$ the expected return cost if a unit of item i is fulfilled through fulfillment node j in period t, then we could extend the base model to include return costs by adding $\sum_{i,j,t}\left(rc_{i,j,t} \times \sum_{o \in O_t} x_{o,i,j}\right)$ to the objective function (2.1).

2.3.3 Literature Classification and Review

Table 2.3 shows the classification of literature on the OAOR following the introduced classification of problem dimensions in Subsection 2.3.1 and 2.3.2. Dimension *External inventory movements* is not considered since no works consider external I-RP or depletion.

The table is sorted to create groups of contributions within the literature that address similar problems, with columns from left to right representing the most important criteria. Indeed, the main criteria to sort literature is the *Order Allocation Timing*, followed by the *Inventory*, *Demand* and *Fulfillment Process Management* problem extensions, since these form the most logical groups of contributions, with groups making the allocation decisions with the same periodicity and optimizing allocation alongside other similar decisions. The other criteria included in the table were also considered when organizing the table but with decreasing importance. The remaining ties are broken using the year of publishing followed by the author's name. Within the table, we highlight with a greyed background the characteristics that make the problem more complex: multiple products, existence of stores and ODC functioning as fulfillment nodes, proactive D-OR/drop-shipping, and inventory, demand, and FPM problem extensions.

We note that every work considers the minimization of some measure of fulfillment costs (order preparation and/or order delivery) and the maximization or minimization of objectives directly related to problem extension decisions (Table 2.2) as the optimization objective. We note a few exceptions:

- In column "Other Objectives", we list objectives that are not directly related to a problem extension and are not fulfillment costs: inventory holding costs (IH), price revenue (PR), and expected customer waiting costs (WT).
- Some works do not consider specific objectives and instead maximize a generic reward function (Balseiro et al. (2023), Amil et al. (2025), Chen et al. (2022), Ma (2023), Wang et al. (2022a), Xie et al. (2025), Andrews et al. (2019) and Bai et al. (2025)). In these works no other objective is considered other than maximizing this function.
- As discussed in Subsubsection 2.3.1.2, most of the literature considers a single objective function. The exceptions are Ardjmand et al. (2018), Wang et al. (2022b) and Das et al. (2023a).
- Asadpour et al. (2019) and Xu et al. (2020) do not consider fulfillment costs in their problem setup, only the cost of lost sales. They do present, however, numerical experiments where fulfillment costs are considered instead.

By analyzing Table 2.3, we first note that most models are single-period within *Batch* problems. Multi-period models imply higher complexity and thus have been explored more

Table 2.3: Classification of OAOR literature

Article	Order allocation timing	*Problem Extensions*				*Other Dimensions (Baseline Problem)*			Other Objectives
		Inventory Management	Demand Management	Fulfillment Process Management	Proactive/ Reactive Unfulfillment/ Drop--Shipping	Fulfillment node types	Number of items and units	Fulfillment node network flexibility	
Acimovic and Graves (2015)	ObO					DC	mP-1U		
Amil et al. (2025)									
Wickman et al. (2023)							mP-mU		
DeValve et al. (2021)					R		1P-1U		
Asadpour et al. (2019)								LC	
Xu et al. (2020)								GCG	
Zhao et al. (2025)							mP-1U	Dual	
Wang et al. (2022a)					P		1P-mU		
Jasin and Sinha (2015)							mP-1U		
Balseiro et al. (2023)									
Ma (2023)									
Chen et al. (2022)							mP-mU		
Xie et al. (2025)									
Gao et al. (2023)						S	1P-1U		
Andrews et al. (2019)							mP-1U		
Bayram and Cesaret (2021)						DC S	1P-1U		PR
Cesaret and Bayram (2020)								Dual	
Akyüz et al. (2022)				F-OC		S	mP-1U		
Wang et al. (2022b)						DC S	mP-mU		
Xiao et al. (2009)			D-CO		R	S	1P-1U	Dual	
Lei et al. (2018)			D-PR		P	DC			
Zhou et al. (2021)						S			
Lei et al. (2021)			D-FR			DC			
Arlotto et al. (2026)		I-I							
DeValve (2023)							mP-1U		
Bai et al. (2025)			D-FR				1P-1U		
Wang and Minner (2024)		I-RP			R		1P-mU		IH
Xu et al. (2009)	SP Batch					DC	mP-mU		
Li and Jia (2019)									
Yang et al. (2022)									
Mahar and Wright (2009)						S	1P-1U		WT IH PR
Mahar et al. (2022)					P		mP-mU		
Das et al. (2023a)				F-CA		DC			
Ardjmand et al. (2018)					R		mP-1U		
Ardjmand et al. (2021)									
Torabi et al. (2015)				F-TR			mP-mU		
Shan and Tian (2022)									
Yang and Li (2023)				F-OR			mP-1U		
Li and Yang (2024)									
Rivière et al. (2021)							mP-mU		
Tang et al. (2023)									
Zhang et al. (2023)					R				
Li et al. (2019)				F-OR F-OC					
Pourahmadi et al. (2023)				F-OR F-TR					
Das et al. (2023b)			D-OR		P	S	1P-1U		
Abdulkader et al. (2018)		I-RP		F-OR		DC	mP-mU		
Bayliss et al. (2020)									
Martins et al. (2020b)									
Martins et al. (2020a)									
Martins et al. (2021)									
Gao et al. (2022)									WT
Wei et al. (2020)	MP Batch					DC	1P-1U	Dual	
Ni et al. (2019)				F-MS		S	1P-mU		
Deng et al. (2024)		I-I			P	DC	mP-mU		IH PR
Govindarajan et al. (2021)						DC S	1P-1U		IH
Jiu et al. (2024)		I-RP			R	DC	mP-mU		
Goedhart et al. (2023)						S	1P-1U		IH PR
Lim et al. (2020)					P	DC			IH
DeValve and Myles (2024)							mP-mU		
Jasin et al. (2022)			D-PR			DC ODC	1P-mU		
Jiu (2022)		I-RP I-TR			R	S ODC	1P-1U		

Papers with *ADL* are: Xie et al. (2025) (ObO), Pourahmadi et al. (2023) (SP Batch), and Wei et al. (2020), Ni et al. (2019), DeValve and Myles (2024), and Jasin et al. (2022) (MP Batch).

recently, the earliest being Ni et al. (2019). *MP Batch* requires solving sequential decision-making problems in batches, rather than applying static optimization methods, as for *SP Batch* problems. Second, we observe that only a few works allow delayed deliveries, that is, in *ObO* problems, to one of the following order arrivals, and in *Batch* problems, to one of the following periods. This may be due to higher difficulty in modeling and solving such problems and the risk of missing the delivery deadline if order allocation is delayed. Third, we notice that most works consider mP and mU orders in *Batch* problems. In contrast, in *ObO* problems, 1P and 1U orders show more prevalence due to the difficulties in considering mU orders in this context. We recommend the reader to check Acimovic and Graves (2015) and Jasin and Sinha (2015) for a more detailed discussion on those issues. We further note that many mP works are restricted to 1U demands, with the authors claiming that having multiple units of the same product in an order is rare. Fourth, we note that the amount of literature considering some orders with fixed source is significant, which shows that authors have addressed not only pure online retail settings but also omnichannel fulfillment networks. Finally, we notice that DM decisions are less common than IM or FPM decisions and are almost exclusive of *ObO* problems, except for Jasin et al. (2022) and Das et al. (2023b), because most DM decisions are pricing and framing decisions made ObO. Extensions with IM decisions are more prevalent in *Batch* problems and with FPM decisions in *SP Batch* problems. Inventory management decisions are usually made between significant periods where multiple orders accumulate. Therefore, it makes sense that they would be considered in *Batch* problems. Moreover, FPM decisions, including routing and consolidation, make the most sense when planning the deliveries of multiple orders, again justifying their prevalence in *Batch* problems.

We next briefly review some papers, focusing in particular on those that introduced new *environment* and *objective* dimensions. We organize this discussion by the *Order Allocation Timing* of the addressed model.

2.3.3.1 Order-by-Order (ObO)

Xiao et al. (2009) were the first to explicitly study online ObO allocation. They considered a retailer who sells a single item in a two-store network with possible drop-shipping. Demand may occur in-store or through one of two online channels: the retailer's own website and through affiliate programs. The retailer's decision consists of dynamically determining which store or drop-shipper to allocate online demand and when to open and close the affiliate programs channel while maximizing profit. Acimovic and Graves (2015) were the first to address the ObO OAOR in a large network context of a retailer selling multiple items to multiple customer regions through multiple fulfillment nodes. In a similar large network, Jasin and Sinha (2015) were the first to explicitly tackle the problem of mP OF, being followed by many authors addressing the same or similar problems (Amil et al. (2025), Ma (2023), Gao et al. (2023) and Wickman et al. (2023)). Zhao et al. (2025) analyzed the problem of mP OF in a small network consisting of a regional and a front DC.

The ObO OAOR is similar to many other online resource allocation problems. Hence, many authors derive policies to solve problem classes that include the OAOR. For instance, Balseiro et al. (2023) defined a general class of problems named dynamic resource con-

strained reward collection problems, which encompasses problems such as network dynamic pricing, multi-secretary, dynamic bidding in repeated auctions with budgets, network revenue management, choice-based revenue management, online matching, and online order allocation. Chen et al. (2022) and Wang et al. (2022a) also introduced fulfillment policies in the context of online resource allocation problems. Xie et al. (2025) studied online resource allocation with order-specific delay.

As discussed in the Baseline Problem, online retailers' fulfillment networks often have limited flexibility. Asadpour et al. (2019), addresses a fulfillment network with a "Long Chain" flexibility. However, Xu et al. (2020) point out that the "Long Chain" design, which assumes that the number of fulfillment nodes and demand locations are equal, is not always applicable in practice. Therefore, they adapted the policy proposed by Asadpour et al. (2019) to a more general type of network called "Connected Networks with Positive Generalized Chaining Gaps", introduced by Shi et al. (2019), which accounts for unbalanced numbers of fulfillment nodes and demand locations.

Some fulfillment contexts may require fulfillment policies to incorporate certain considerations, for example, preference for operating under a local no-holdback fulfillment constraint, where each customer order must be fulfilled from its primary fulfillment node (implicitly, the closest) if it has enough inventory, which is addressed by DeValve et al. (2021). Akyüz et al. (2022) addresses a context where the consolidation of mP orders is mandatory before final delivery. Bayram and Cesaret (2021) introduces an omnichannel context with multiple stores and one fulfillment center, where the online channel and each physical store externally set their own prices. Cesaret and Bayram (2020) consider a similar problem but with a single store and a single fulfillment center and with potential customer switching between online and offline channels. Wang et al. (2022b) considers a similar omnichannel context, but considering the existence of multiple fulfillment options (home delivery and pick-up in-store, either urgent or appointed).

A few ObO studies have jointly optimized order allocation with other operational decisions, mostly pricing. The first study to address the problem of joint pricing and order allocation of a single item in an online retail context was Lei et al. (2018). Later, Lei et al. (2021) extended this work to consider the decision of framing multiple items, where each item is assigned a display or ranking position with a particular price. Furthermore, Zhou et al. (2021) studied the problem of pricing and order allocation in an omnichannel context, where each item can have different prices in each physical store and on the online channel. Another decision that has been recently been jointly optimized with ObO allocation is I-I ordering and placement (DeValve (2023) and Arlotto et al. (2026)). Bai et al. (2025), who also optimize the inventory ordering and placement decisions, introduce a model where each customer is proposed a specific delivery deadline (e.g., 1-day, 2-day) alongside a fulfillment node choice, which the customer may accept or reject (the latter resulting in lost demand).

2.3.3.2 Single Period Batch

Xu et al. (2009) were the first to address the Single Period Batch OAOR. Afterward, other works focused on exclusively optimizing order allocation decisions in batches, introducing

new problem features, such as additional fulfillment objectives or problem constraints. Mahar and Wright (2009) addressed an omnichannel context to minimize expected inventory holding and back-ordering costs, transportation and handling costs, and customer waiting costs. Later, Mahar et al. (2022) extended the aforementioned work to the mP context. Yang et al. (2022) proposed a graph-based Machine Learning (ML) model to optimize the problem of order allocation with item perishability. Das et al. (2023b) analyzed the problem of dynamically accepting orders which were later allocated to fulfillment nodes in batches.

Order batches are often used to optimize fulfillment node allocation along with other operational decisions. Ardjmand et al. (2018) and Ardjmand et al. (2021) addressed the joint problem of optimizing order allocation with F-CA. Later, Das et al. (2023a) solved a similar joint problem of order allocation and cartonization, but considering a variable carton size, with the goal of minimizing shipping costs and packaging volume.

Torabi et al. (2015) jointly determines allocation and shipment decisions from fulfillment nodes to customers with reactive transshipments between fulfillment nodes. Pourahmadi et al. (2023) address a similar problem considering scenarios with varying delivery parameters (lead time and costs). Li and Jia (2019) optimizes the decisions of allocating orders to fulfillment nodes and, subsequently, the transportation to delivery stations within order-specific time-windows. Li et al. (2019) optimizes order allocation with delivery planning in a two-echelon network. In the upper echelon, demand is assigned to fulfillment nodes and then routed to DC using urban vehicles for consolidation. In the lower echelon, consolidated orders are routed from DC to customers using street vehicles. Rivière et al. (2021) consider a similar problem in a simpler single-echelon network, with the goal of routing demand from stores to customers. Recently, Tang et al. (2023), Yang and Li (2023), Zhang et al. (2023), and Li and Yang (2024) all address the problem of allocating demand and routing it from fulfillment nodes to customers, with some context-specific variations. Shan and Tian (2022) study the context of online supermarkets, where customers make orders with multiple items that are split into sub-orders according to warehouse item availability. Sub-orders are then consolidated using reactive transshipments between warehouses before the final delivery.

In the same context of optimizing order allocation with order delivery decisions, Abdulkader et al. (2018) introduced the omnichannel vehicle routing problem (OVRP). The OVRP considers a homogeneous set of vehicles that, starting from a single depot, simultaneously replenishes inventory at retail stores and uses store inventory to fulfill the demand of online customers before returning to the depot, where the inventory replenished is independent of the inventory used for demand fulfillment, due to requiring further in-store processing. Many subsequent works have addressed the OVRP (Bayliss et al. (2020), Gao et al. (2022), Bayliss et al. (2020), Martins et al. (2020b), Martins et al. (2020a), Martins et al. (2021) and Gao et al. (2022)).

2.3.3.3 Multi Period Batch

Literature featuring the MP Batch OAOR has become more prevalent in recent years, as the field of fulfillment optimization in online retail has matured. In Ni et al. (2019), the

authors consider multiple delivery modes (own fleet and two crowdsourcing-based modes) and determine order allocation to stores jointly with delivery mode selection. Wei et al. (2020) focused on a two-warehouse network where the goal was to determine, at each decision period, which orders to delay and to which warehouse to allocate the remaining orders. Govindarajan et al. (2021) and Deng et al. (2024) studies the problem of optimizing I-I levels and multi period batch order allocation.

The first to jointly optimize order allocation with I-RP decisions was Lim et al. (2020), with the goal of minimizing shipping and inventory-related costs (replenishment and holding) in a single-echelon distribution network. Jiu (2022) addressed a two-echelon network with DC and stores, where the replenishment of both facility types is optimized jointly with the allocation of online orders to both DC and stores, while Jiu et al. (2024) focuses on the same decisions on a two-echelon network with rear and front DC. DeValve and Myles (2024) combines the problem of I-RP with the order allocation policy of backlogged demand. Goedhart et al. (2023) formulate the OAOR with I-RP as an Markov Decision Process (MDP). Jasin et al. (2022) address the problem of joint I-RP, D-PR, and order allocation, where replenishment and pricing decisions are made at the start of each period, and order allocation decisions at the end, after demand is observed.

2.4. Allocation Policy

In the tutorial by Acimovic and Farias (2019), the authors highlight four mechanisms for optimizing the OAOR. While their aim was not an exhaustive analysis of all the literature, these four mechanisms effectively encompass most reviewed papers (c.f. Table 2.4). In the following discussion, we delve into these four mechanisms and complement them with other unidentified mechanisms. Then, we look at four other dimensions, which have to do with the allocation policy itself and the information it uses.

2.4.1 Optimization mechanism

Batching orders (Batch) One of the strategies identified is to batch orders, which consists of approximating an ObO problem to SP or MP Batch problems, as discussed in Subsubsection 2.3.1.1.

In SP Batch problems with deterministic demand, static solution methods can be used. Classical optimization techniques for static problems, used in the context of the OAOR, ranges from exactly solving a mixed-integer linear program (e.g., Mahar and Wright (2009)) to utilizing heuristics, such as Benders decomposition (e.g., Torabi et al. (2015)), improvement heuristics (e.g., Xu et al. (2009)), and greedy heuristics (Martins et al. (2020b)); and metaheuristics, such as adaptive large neighborhood search (e.g., Li et al. (2019)), ant colony optimization (e.g., Abdulkader et al. (2018)), and genetic algorithms (e.g., Ardjmand et al. (2018)). Other than classic optimization methods, Yang et al. (2022) proposes applying supervised learning utilizing a dataset of previously made allocations.

In MP Batch problems, dynamic optimization methods need to be applied, as the problem is a sequential decision-making problem. These works combine batching orders with other mechanisms, which we will discuss later.

Adding opportunity costs to immediately realized rewards (Opportunity Cost) Another strategy is to devise a policy that utilizes the following structure:

$$\max_{u \in U_t} \{R(O_t, u) + V(inv(\boldsymbol{s}_t, O_t, u), t)\} \tag{2.6}$$

Equation (2.6) is based on the Bellman optimality equation. $R(O_t, u)$ represents the reward earned for orders arriving between period $t-1$ and t earned by making allocations u, out of the set of feasible allocations U_t (that reflect the constraints on inventory availability and demand fulfillment for period t in equations 2.2 and 2.3, respectively), for the set of orders O_t. $inv(\boldsymbol{s}_t, O_t, u)$ represents the transition function to the new inventory state $\boldsymbol{s}_{t+1}$ that evolves according to the current inventory state $\boldsymbol{s}_t$ and the fulfillment actions u taken to fulfill orders O_t. Assuming that the reward $R(O_t, u)$ is easy to compute, the crux of this strategy is to approximate the opportunity costs, $V(\boldsymbol{s}_{t+1}, t)$ (more generally referred to as value function in the dynamic programming community).

Acimovic and Graves (2015) proposed to approximate the opportunity cost by utilizing the dual value of the inventory constraint of an appropriately constructed linear program, whereas Ni et al. (2019) approximates the entire equation $R(O_t, u) + V(inv(\boldsymbol{s}_t, O_t, u), t)$ with a single mixed-integer linear model. Both of these approaches require constructing a deterministic lookahed (see next mechanism) to estimate the opportunity costs. Goedhart et al. (2023) uses a one-step policy improvement to approximate the value function. To circumvent the need for demand forecasts (as is required in Acimovic and Graves (2015), Ni et al. (2019), and Goedhart et al. (2023)), Andrews et al. (2019) utilizes a primal-dual algorithm where the dual variables, representing the opportunity costs, do not require forecasts for their estimations, relying instead on learning based on past allocation decisions.

Utilizing a deterministic lookahed (DL) Another approach is approximating the original problem to a deterministic one. Consider a simple 1P, 1U ObO problem with only order allocation decisions. Assume (a) r to be an order delivery region, (b) up to one order arrives in each period, with λ_r^i being the probability that an order for item i to be delivered in region r arrives in a period, such that $\sum_j^i \lambda_r^i \leq 1$, (c) $c_{j,i,r}$ to be the cost of shipping one unit of item i from fulfillment node j to region r, and (d) $x_{j,i,r}$ to represent the proportion of times item i should be allocated to fulfillment node j. Consider another problem that minimizes the following equation:

$$\sum_{i,j,r} T \lambda_r^i \left(c_{j,i,r} \times x_{j,i,r} \right) \tag{2.7}$$

subject to constraints ensuring demand fulfillment and inventory availability. This problem is a deterministic approximation of the original ObO problem. The solution of this deterministic approximation is then used to make order allocation decisions. A particular case of this approach often used in OF is utilizing the order allocation variables ($x_{j,i,r}$ in the example above) as a probability distribution to sample order allocation decisions

as orders arrive, effectively randomizing OF (**DL-Randomize**). These methods are supported by their attractive asymptotic properties (Acimovic and Farias, 2019). Randomized solution approaches are common not only in the OAOR but also in several revenue management problems, as noted by Chen et al. (2022), Wang et al. (2022a), and Balseiro et al. (2023). We highlight Jasin and Sinha (2015) for introducing rounding schemes specifically designed for mP orders, which were later improved by Ma (2023) and Amil et al. (2025). The latter combines ideas of Randomized Fulfillment (RF) with prophet inequalities and subgradient methods. Bai et al. (2025) add an additional rollout step, based on Value Function Approximations, to improve the decisions proposed by their randomized policy.

Setting Partitions (Inventory/Order/Cost/Delivery P) Under this strategy, the order allocation policy defines partitions (or thresholds) that reserve inventory for future (implicitly, more important) allocations. Assuming $limit_{i,j,t}$ to be the maximum amount of demand of item i that can be allocated to fulfillment node j in period t, then adding the constraint:

$$\sum_{o \in O_t} x_{o,i,j} \leq limit_{i,j,t} \ \forall i, j, t \tag{2.8}$$

to the baseline model in Subsection 2.3.1 transforms the model into an inventory threshold policy. Similarly to this policy, DeValve et al. (2021) assigns a threshold for each fulfillment node, such that if the fulfillment node's inventory level falls below the threshold, only local fulfillment is permitted (fulfillment of demand arising from the node's region). Alternatively, it considers a threshold for each fulfillment node-region pair, which prohibits fulfillment if inventory depletes below the threshold. Govindarajan et al. (2021) and Zhao et al. (2025) define similar inventory thresholds to protect inventory for store or other types of local fulfillment. Xiao et al. (2009) proposes an optimal policy for a simple one-item two-store network defined by thresholds in the two-dimensional inventory space. Authors have also proposed other partition-based policies that are not based on inventory. Specifically, Wei et al. (2020) and Bayram and Cesaret (2021) have proposed policies based on thresholds that depend on the delivery-time urgency of yet-to-be-allocated orders and the OF cost for each fulfillment node, respectively. Gao et al. (2023) uses order thresholds on the number of accepted online orders and then uses a randomized scheme to determine order allocation decisions.

Other We now briefly discuss other solution methods that are used less often.

- **Lagrangian Relaxation (LR)** – A form of approximation found in literature, an alternative to the deterministic approximation required for RF, was proposed by Jasin et al. (2022). The authors use LR to create an approximate problem with reduced interdependency between variables. A policy based on the optimal solution of the relaxed problem is derived to solve the original problem.
- **Dynamic Objective Function (Dynamic OF)** – Akyüz et al. (2022) considers an ObO allocation problem with a concave cost function whose parameters marginally

decrease as allocations are made. This allows the approximation of the original dynamic stochastic problem to multiple static deterministic problems, connected by the "shifts" in the objective function. The authors propose solving the problem using an iterative matheuristic based on the solution of the set covering model and local search.

- **Load Deviation** – Asadpour et al. (2019) and Xu et al. (2020) propose a policy based on the load deviation, i.e., the deviation between the expected allocation to fulfillment nodes had they had infinite inventory and the actual allocations. These policies allocate orders to minimize the load deviation to protect the inventory of nodes whose expected demand exceeds its inventory.

- **Assigned Backlog Base-Stock (ABBS)** – DeValve and Myles (2024), in an MP Batch problem with demand backlog, consider a policy that combines a base-stock level for I-RP with backlog assignment (to the maximum extent) in every period. A stochastic problem is utilized to optimize this policy.

- **Intuition-based policy (Intuitive)** – Wang et al. (2022b) develop a policy with a distinct structure based on "while" and "if" conditions. The authors based this policy on the trade-off between customer satisfaction and fulfillment cost across online and offline channels.

- **Actor-critic** – Wang and Minner (2024) utilizes Proximal Policy Optimization, an Actor-critic method that utilizes two Neural Network (NN), one working as the actor, which represents the policy making decisions, and another working as the critic, which learns a value function to help the critic learn.

2.4.2 Decision-making policy class and approximation/solution technique

ObO and *MP Batch* problems belong to a larger class of sequential decision-making problems under uncertainty, encompassing several research communities, from stochastic optimization to Reinforcement Learning (RL). Powell (2022) proposes a framework that organizes policies for those problems in four main classes that we now describe.

Policy Function Approximation (PFA) PFAs are policies that consist of analytical functions that, explicitly or implicitly, map problem states to actions without requiring the solution of an optimization problem. For example, Load Deviation policies, the ABBS policy by DeValve and Myles (2024), and the intuition-based policy by Wang et al. (2022b) are PFAs. These could take, however, many other forms, such as lookup tables, where each state or cluster of states is mapped to an action or any parametric or nonparametric representation.

Cost Function Approximation (CFA) CFAs are optimization models that modify the objective function and/or constraints with a parameterization. Policies based on partitions are CFAs, since they add a constraint with a parameter that needs to be optimized. The

iterative metaheuristic in Akyüz et al. (2022) is also a form of CFA since the objective function is parameterized by its shifting function.

Value Function Approximation (VFA) VFAs are related to approximate dynamic programming, where the goal is to learn how to estimate value function, thus corresponding to the "Opportunity Cost" optimization mechanism. Consequently, policies incorporating opportunity costs to immediately realized rewards are a form of VFA. However, in the OAOR, strategies for VFA have been limited to using a linear or mixed-integer program for estimation (Acimovic and Graves (2015), Ni et al. (2019)), value iteration (Goedhart et al., 2023) and a primal-dual approach (Andrews et al., 2019). Other VFA approaches could also be used, for example, Q-learning.

Direct Lookahead Approximation (DLA) DLAs are policies that directly look into the future, considering possible scenarios. Policies relying on randomizing OF and LR are DLAs, since the underlying mechanism consists of solving a simplified problem (deterministic approximation and LR, respectively) that estimates the entire problem horizon over multiple periods. The same is true for the two-phase stochastic programs in Lim et al. (2020) and Jiu (2022), in which a lookahead is used to predict order allocation when making pre-demand arrival decisions (I-RP). Generically, DLAs are policies that somehow embed into an optimization problem estimated future states and decisions without relying on value functions.

Often, solution methods utilize more than one of the four aforementioned policy classes. Gao et al. (2023) combines RF (a form of DLA) with booking limits (a form of CFA), whereas Wang and Minner (2024) utilizes Proximal Policy Optimization, a RL Technique featuring two Deep NN, the *critic* functioning as a VFA and and *actor* functioning as a PFA.

Static Approximation (SA) We also add SA as the fifth policy class to include all policies used to solve *SP Batch* problems.

2.4.3 Future demand information and learning

Some policies require forecasting future demand, while others do not. In practice, this is an important distinction because a reliable forecast is often unavailable. Generally, policies that rely on batching orders do not require a demand forecast (c.f. Table 2.4). Those policies only consider the demand that has effectively arrived during the current delay period, leaving demand arriving in the next period to be considered only in the subsequent batch.

Contrarily, policies that utilize RF and opportunity costs usually require demand forecasts, which are used as input to construct the deterministic approximation required for RF and to estimate the opportunity costs, respectively. The exception is Andrews et al. (2019), which does not rely on demand forecast for opportunity cost estimation, but instead on learning them using a primal-dual approach.

We note that some policies with delayed fulfillment utilize forecasts. E.g., Mahar and Wright (2009) and Mahar et al. (2022) require forecasts for in-store demand. Lim et al. (2020) and Jiu (2022) use the expected demand when making inventory decisions before demand arrival and subsequent order allocation. Ni et al. (2019) requires a forecast for opportunity cost estimation, while Xie et al. (2025) needs it to solve its online linear program.

Regarding policies based on partitions/thresholds, the need for forecasts depends on how those are derived. Govindarajan et al. (2021) computes its inventory thresholds based on the I-I policy, which requires demand forecasts. Bayram and Cesaret (2021) also needs a forecast to compute its cost thresholds. Policies based on load deviation do not require forecasts since those policies are based on actual demand and fulfillment conditions.

Within the context of the OAOR, there is still a limited amount of policies with learning, i.e., using information acquired from past decisions to improve future decision-making. Two distinct settings exist: online, where the learning occurs while the policy is deployed, such as in Andrews et al. (2019) and Wang and Minner (2024), and offline, where the learning occurs in a simulation before policy deployment. Using the latter, Wei et al. (2020) proposes using simulation-based approaches to learn the thresholds for its policies, Yang et al. (2022) proposes a supervised learning approach, where past optimal allocation decisions are used to learn the fulfillment policy, and Wickman et al. (2023) propose using what they term a *value lookahead strategy* which consists in using simulated samples to learn the value function, which are then used for order allocation.

2.4.4 Literature Classification and Discussion

Table 2.4 presents a classification of the literature's policies following the dimensions discussed in this Section, plus the "Policy Type", which pertains to the model used in each paper to approximate the policy function, value function, and direct lookahead model for the PFA, VFA and DLA classes, respectively. In the remaining cases (CFAs and SAs), it pertains to the method used to solve the paper's model.[§] The articles are sorted as in Table 2.3. Besides the *Policy Classification* columns, the *Order Allocation Timing* and *Problem Extension* columns from the *Problem classification* are included to help to connect Table 2.3 to Table 2.4. Some works solve their model using an optimal solution method (Cesaret and Bayram (2020) and Xiao et al. (2009)) using Dynamic Programming, which we do not classify under any particular *Policy Class*.

We can observe that, beyond *SP Batch*, that exclusively use static optimization methods, DLAs (namely Randomize and Direct Lookahead policies) are dominant with 17, 7, 6, and 4 works for VFAs, CFAs and PFAs, respectively. Some works hybridize multiple approaches, with Gao et al. (2023) hybridizing PFA with DLA, Acimovic and Graves (2015), Ni et al. (2019) and Bai et al. (2025) DLA with VFA, and Wang and Minner (2024) PFA with VFA. Regardless, Powell (2022) notes that it is often unclear which policy class is better in a particular context, often being up to researchers to make comparisons. In

[§] "LP" – linear program; "MIP" – mixed linear-integer program; "DP" – dynamic programming; "VFA" – Value Function Approximation; "RF" – Randomized Fulfillment; "GA" – genetic algorithm; "ALNS" – adaptive large neighbourhood search; "SA" – simulated annealing; "MDP"– Markov Decision Process; "VNS" – Variable Neighbourhood Search; "RTT" – Randomized Tabu Thresholding

Table 2.4: Classification of OAOR literature according to its policy

Article	*Problem Classification*		*Policy Classification*				
	Order Allocation Timing	**Problem Extension**	**Optimization mechanism**	**Policy Class**	**Policy Type**	**Future Demand**	**Learning**
Acimovic and Graves (2015)	ObO		Opportunity Cost + DL	VFA + DLA	LP based VFA	x	
Amil et al. (2025)			DL-Randomize	DLA	LP based RF		
Wickman et al. (2023)			Opportunity Cost	VFA	Value Function based policy		Offline
DeValve et al. (2021)			Inventory P	CFA	Spillover limit policies		
Asadpour et al. (2019)			Load Deviation	PFA	Modified greedy policy		
Xu et al. (2020)							
Zhao et al. (2025)			Inventory P	CFA	Threshold based policy		
Wang et al. (2022a)			DL-Randomize	DLA	LP based RF	x	
Jasin and Sinha (2015)							
Balseiro et al. (2023)							
Ma (2023)							
Chen et al. (2022)							
Xie et al. (2025)			DL		LP based heuristic		
Gao et al. (2023)			Order P + DL-Randomize	CFA + DLA	RF based on Booking limit		
Andrews et al. (2019)			Opportunity Cost	VFA	Primal-Dual algorithm		Online
Bayram and Cesaret (2021)			Cost P	CFA	Cost threshold based policies	x	
Cesaret and Bayram (2020)			Optimal	-	DP		
Akyüz et al. (2022)		FPM	Dynamic OF	CFA	Iterative metaheuristic		
Wang et al. (2022b)			Intuitive	PFA	Simple heuristic		
Xiao et al. (2009)		DM	Optimal	-	DP	x	
Lei et al. (2018) [1]			DL-Randomize	DLA	LP based RF		
Zhou et al. (2021) [1]							
Lei et al. (2021) [1]							
Arlotto et al. (2026) [2]		IM					
DeValve (2023)							
Bai et al. (2025)		IM + DM	Opportunity Cost + DL-Randomize	VFA + DLA	LP based RF with VFA based rollout		
Wang and Minner (2024)		IM	Actor-critic	VFA + PFA	Proximal Policy Optimization		Online
Xu et al. (2009)	SP Batch	FPM	Batch	SA	Improvement heuristics		
Li and Jia (2019)					Benders Decomposition		
Yang et al. (2022)					Graph attention mechanism		Offline
Mahar and Wright (2009) [3]					MIP	x	
Mahar et al. (2022) [3]							
Das et al. (2023a)					Ant Colony Optimization		
Ardjmand et al. (2018)					NSGA-II		
Ardjmand et al. (2021)					Four-stage heuristic; GA; SA		
Torabi et al. (2015)					Benders Decomposition		
Shan and Tian (2022)							
Yang and Li (2023)					Tabu search and memetic algorithm		
Li and Yang (2024)					VNS and RTT		
Rivière et al. (2021)					GA		
Tang et al. (2023)					Branch-and-cut		
Zhang et al. (2023)					Local Search Heuristic		
Li et al. (2019)					ALNS		
Pourahmadi et al. (2023) [4]					Robust scenario-based SP		
Das et al. (2023b)		DM			LP		
Abdulkader et al. (2018)		IM + FPM			Two-phase heuristic; Multi-ant colony		
Bayliss et al. (2020)					Two-phase heuristic		
Martins et al. (2020b)					Three-stage greedy heuristic		
Martins et al. (2020a)					Simheuristic		
Martins et al. (2021)					Biased-random algorithm		
Gao et al. (2022)					GA		
Wei et al. (2020)	MP Batch		Batch + Delivery P	CFA	Threshold based heuristics		Offline
Ni et al. (2019)		FPM	Batch + Opportunity Cost + DL	VFA + DLA	MIP based VFA	x	
Deng et al. (2024)		IM	Batch + DL	DLA	LP based heuristic		
Govindarajan et al. (2021)			Batch + Inventory P	CFA	Threshold based policy		
Jiu et al. (2024)			Batch + DL	DLA	Robustness based Benders decomposition		
Goedhart et al. (2023)			Batch + Opportunity Cost	VFA	MDP decomposition with value iteration		
Lim et al. (2020)			Batch + Two-phase RO	DLA	Two-phase RO		
DeValve and Myles (2024)			Batch + ABBS	PFA	ABBS policy		
Jasin et al. (2022)		IM + DM	Batch + LR	DLA	Lagrangian approximation--based policy		
Jiu (2022)		IM	Batch + Two-phase RO		Two-phase RO		

[1] Demand depends on other decisions
[2] Also has a score-based alternative, where, after solving an LP, instead of randomizing fulfillment, the fulfillment option with the highest "probability" is chosen
[3] Only in-store demand forecast
[4] Includes multiple scenarios with varying lead times and costs

most works that we surveyed, authors generally only compared their proposed policies with other simpler policies proposed by themselves (e.g., a variation of a simple myopic policy) and/or an upper/lower bound policy (e.g., optimal policy). When comparisons are made, they are generally restricted. For example, Ma (2023) and Amil et al. (2025) show how their methods improve over Jasin and Sinha (2015), but all are based on the same optimization mechanism. DeValve et al. (2021) compare its method against that of Acimovic and Graves (2015), showing that a simpler policy, based on setting spill-over limits, can have a performance close to that of a more complex LP based policy. Asadpour et al. (2019) compare its method with the one proposed by Jasin and Sinha (2015), showing better performance across all tested parameters (but on a single product setting, whereas Jasin and Sinha (2015) policy was designed specifically to deliver good performance with mP orders), and Acimovic and Graves (2015), where the policy proposed by Asadpour et al. (2019) shows improved performance when the number of fulfillment nodes is relatively small, and negligible differences when the problem horizon (the number of orders) is large. Outside these examples, the comparison of methods is limited.

2.5. Research Opportunities

Building upon the insights gained from the previous sections, we now discuss potential future directions for the evolution of the OAOR literature. This discussion will revolve around unexplored or under-explored problem settings and solution methods. By identifying these areas, we aim to shed light on the untapped opportunities for further research and development in the field of the OAOR optimization.

2.5.1 Problem settings

Order-specific Allocation Delay Table 2.3 highlights that most works on delayed order allocation do not consider a delay, such that every order is allocated in the period immediately following its arrival. We conjecture that a more precise way would be to consider the order-specific delivery deadlines with the available fulfillment options to determine the optimal order-specific ADL length by determining each order's estimated-to-ship date (Xu et al., 2009). For example, Xu et al. (2009) immediately allocate demand upon arrival and periodically improves allocation decisions by performing a local search on the incumbent allocation. This ensures that the improvement procedure maintains a feasible solution and monotonically decreases costs. It allows fulfillment to be delayed until each order's estimated-to-ship date. Another approach would be to only allocate each order at the estimated-to-ship date, allowing to still gain knowledge of demand that occurred between the order's arrival and the estimated-to-ship date while removing the need to do the improvement procedure. Nevertheless, this could prove more complex and less reliable if an unexpected event forced allocation to occur before. More policies like that of Xu et al. (2009) incorporating opportunity cost estimation and learning could be developed to improve this further.

Product substitution Some online and omnichannel retailers allow the substitution of out-of-stock products with other similar ones. Product substitutions is thus a mechanism to avoid lost demand and (at least partially) customer dissatisfaction. Considering substitution in the context of OAOR optimization can be extremely beneficial, as it could be used, for example, as a mechanism to protect inventory for customers who do not allow or are dissatisfied by substitution. However, this has not been considered in the context of the OAOR.

Delivery delays, errors and order returns Operational realities (e.g., congestion at fulfillment nodes or delivery processes) often introduce uncertainty in delivery times which can negatively impact customer experience. Furthermore, order returns and the subsequent reverse fulfillment process are relevant to retailers, more so in online contexts (e.g, BORIS), where the return rate is particularly high. Therefore, considering that operational disruptions can occur at fulfillment nodes (leading to picking delays, miss-picking, or wrong packaging) and/or at shipping routes (leading to transportation delays or damage), reducing the amount of delayed deliveries, delivery errors, and returned inventory becomes imperative. This issue has not been considered in the literature yet, presenting as an interesting research avenue.

Inventory movements (replenishment and depletion) from external sources In the OAOR literature published so far, a single agent controls the entire inventory network, effectively controlling inventory depletion and potential replenishment. However, in practice, external factors can deplete inventory: perishability, damage in storage and handling, and even actions by external agents, such as those by third-party vendors in online marketplaces (e.g., selling inventory through a distinct sales channel). External factors leading to inventory depletion and replenishment may change the urgency in allocating certain inventory units, since inventory may be lost without any benefit to the allocating agent. Consequently, investigating fulfillment settings and policies considering those issues constitutes an interesting research direction.

2.5.2 Solution Methods

Multi-objective optimization Several objectives have been addressed by the literature on the OAOR, namely those related to the costs of the fulfillment process, customer satisfaction and loss of goodwill, and the profit generated by other decisions such as pricing and revenue management, often within the same work. Other objectives, for example, the fair allocation of demand to vendors in the context of online marketplaces, as alluded to in Section 2.1, are yet to be addressed.

Furthermore, most literature aggregates multiple objectives into a single function to be minimized or maximized, often by translating non-monetary objectives, for example, those related to customer satisfaction and loss of goodwill, to costs or rewards, which requires user subjectivity. Some authors propose alternative ways to consider multiple objectives in optimization problems. Namely, Ardjmand et al. (2018) proposes maintaining separate objective functions and utilizing approaches based on Pareto optimality principles. We

suggest that future research considers similar methods in the context of order allocation optimization.

Reinforcement learning based optimization approaches In Section 2.4, we discussed how most of the OAOR optimization policies do not utilize learning mechanisms. The lack of thereof limits policies' ability to adapt to specific fulfillment contexts. Learning approaches should be particularly useful in highly stochastic and non-stationary large-scale fulfillment settings, where analytical approaches are usually intractable.

We particularly highlight RL as a set of interesting approaches to incorporate learning into the optimization of the OAOR. Generally speaking, in RL an agent utilizes feedback from its actions to improve (reinforce) the policy that it uses to make decisions. Any of the four policy classes for sequential decision-making (PFA, CFA, VFA and DLA) can admit RL policies. We forward the readers to Yan et al. (2022), a recent review of RL approaches and implementations in logistics and supply chain management problems. In particular, we highlight *Resource allocation* and *Inventory control* as the fields to which the OAOR is related to, which have had significant contributions in recent years and can serve as a basis for RL applications for the OAOR. Recently, Wang and Minner (2024) proposed a Deep Reinforcement Learning (DRL) algorithm based on the proximal policy optimization algorithm to solve the OAOR, showing promising results when compared to a myopic policy. We expect more contributions in the area of RL and DRL in the near future.

Policy benchmarking Towards the end of Section 2.4, it becomes evident that the circumstances favoring the effectiveness of specific policies are still unclear. To address this, it is essential to: (a) in dynamic scenarios like ObO and MP Batch, identify the characteristics influencing the performance of myopic fulfillment policies in relation to optimal ones, enabling the recognition of problem types requiring more sophisticated non-myopic methods; (b) prompt authors to compare their proposed fulfillment policies not only with simpler benchmarks (simpler policies proposed by themselves) but also with established alternatives in the literature, as comparisons often involve simpler policies, leaving the significance of their contribution uncertain; and (c) undertake a comprehensive comparative study to understand the conditions under which certain policy classes outperform others. The latter effort can involve several research questions in itself, for example, understanding the factors that make estimating a value function challenging or a lookahead approximation inaccurate, providing insights into when PFAs or CFAs are preferable. Addressing these issues would aid researchers in pinpointing settings with the highest potential for improvement and assist practitioners in selecting the most suitable existing policies for their specific contexts.

2.6. Conclusion

The online order allocation problem has arisen alongside the advent of online and omnichannel retail. It is the operational problem that retailers face of allocating order-line

requests to inventory at one of multiple fulfillment nodes in the retailer's network. Although a seemingly simple decision, it influences the supply process, by, alongside inventory decisions, determining the retailer's network inventory state, the demand process, since presented fulfillment options and the success of the post-allocation fulfillment process influence customer behaviour, and the fulfillment process itself, since the allocated fulfillment nodes act as the starting fulfillment point.

Due to the breadth of online and omnichannel retail contexts appearing over the years, and intersection of order allocation with other operational tasks, literature on the OAOR has recently experienced significant and distinct contributions. However, there is a lack of standard over which to analyze, systematize, and compare different contributions on the problem.

To that end, our work provides the first systematic literature review, capturing all works related to the OAOR. In total, 61 articles were selected, with over 75% published from 2020 onwards, showing the recent relevance of the literature. Our work systematizes the literature to propose the first framework to classify OAOR articles according to problem dimensions, with dimensions related to the baseline problem that only consider the order allocation decision and others related to extensions with other jointly optimized decisions and different optimization methods based on extending the four optimization mechanisms introduced by Acimovic and Farias (2019) and the policy classes for sequential decision making introduced by Powell (2022).

Our work concludes by providing insights into potential future research in terms of new and under-addressed problem variants, such as order-specific allocation demand, product substitution, delivery delays and errors, order returns, and external inventory movements; new solution methods, namely those focused on multi-objective optimization and RL; and a call for the importance of undertaking comparisons between different fulfillment policies established in the literature for benchmarking purposes, which has been lacking.

Bibliography

M. M.S. Abdulkader, Yuvraj Gajpal, and Tarek Y. ElMekkawy. Vehicle routing problem in omni-channel retailing distribution systems. *International Journal of Production Economics*, 196:43–55, 2018. doi: 10.1016/J.IJPE.2017.11.011.

Jason Acimovic and Vivek F. Farias. *The Fulfillment-Optimization Problem*, chapter 9, pages 218–237. INFORMS, 2019. doi: 10.1287/educ.2019.0199.

Jason Acimovic and Stephen C. Graves. Making better fulfillment decisions on the fly in an online retail environment. *Manufacturing & Service Operations Management*, 17(1): 34–51, 2015. doi: 10.1287/MSOM.2014.0505.

Niels A.H. Agatz, Moritz Fleischmann, and Jo A.E.E. van Nunen. E-fulfillment and multi-channel distribution - a review. *European Journal of Operational Research*, 187(2): 339–356, 2008. doi: 10.1016/j.ejor.2007.04.024.

Deniz Aksen and Kemal Altinkemer. A location-routing problem for the conversion to the "click-and-mortar" retailing: The static case. *European Journal of Operational Research*, 186(2):554–575, 2008. doi: 10.1016/J.EJOR.2007.01.048.

M. Hakan Akyüz, Ibrahim Muter, Güneş Erdoğan, and Gilbert Laporte. Minimum cost delivery of multi-item orders in e-commerce logistics. *Computers & Operations Research*, 138:105613, 2022. doi: 10.1016/J.COR.2021.105613.

Ayoub Amil, Ali Makhdoumi, and Yehua Wei. Multi-item order fulfillment revisited: LP formulation and prophet inequality. *Management Science*, 71(12):9917–9935, 2025. doi: 10.1287/mnsc.2023.00357.

John M. Andrews, Vivek F. Farias, Aryan I. Khojandi, and Chad M. Yan. Primal–dual algorithms for order fulfillment at Urban Outfitters, Inc. *INFORMS Journal on Applied Analytics*, 49(5):355–370, 2019. doi: 10.1287/INTE.2019.1013.

Ehsan Ardjmand, Omid Sanei Bajgiran, Shakil Rahman, Gary R. Weckman, and William A. Young. A multi-objective model for order cartonization and fulfillment center assignment in the e-tail/retail industry. *Transportation Research Part E: Logistics and Transportation Review*, 115:16–34, 2018. doi: 10.1016/J.TRE.2018.04.005.

Ehsan Ardjmand, William A. Young II, and Shaki Rahman. Order cartonization and fulfillment center assignment in the retail industry. *Journal of International Business Disciplines*, 16(2):89–126, 2021.

Alessandro Arlotto, Irem Nur Keskin, and Yehua Wei. Online demand fulfillment problem with initial inventory placement: A regret analysis. *Operations Research*, 74(3):1153–1164, 2026. doi: 10.1287/opre.2024.0983.

Arash Asadpour, Xuan Wang, and Jiawei Zhang. Online resource allocation with limited flexibility. *Management Science*, 66(2):642–666, 2019. doi: 10.1287/MNSC.2018.3220.

Sanam Azadiamin and Dale T. Masel. Last mile delivery route planning for grocery stores. In *IIE Annual Conference. Proceedings*, pages 1106–1111. Institute of Industrial and Systems Engineers (IISE), 2021. URL https://www.proquest.com/docview/2560888720.

Yicheng Bai, Paat Rusmevichientong, and Huseyin Topaloglu. Joint placement, delivery promise, and fulfillment in online retail. *Management Science*, 71(11):9171–9192, 2025. doi: 10.1287/mnsc.2023.01726.

Santiago R. Balseiro, Omar Besbes, and Dana Pizarro. Survey of dynamic resource-constrained reward collection problems: Unified model and analysis. *Operations Research*, 72(5):2168–2189, 2023. doi: 10.1287/opre.2023.2441.

Rajendra Baraiya and Rohit Kapoor. Shared capacity routing problem for buy-online-pickup-in-store order fulfillment. In *Proceedings of the International Conference on Industrial Engineering and Operations Management*, pages 6856–6857, 2021. URL https://index.ieomsociety.org/index.cfm/item/41897.

Opher Baron, Andre A. Cire, and Sinem K. Savaser. Decentralized online order fulfillment in omni-channel retailers. *Production and Operations Management*, 33(8):1719–1738, 2024. doi: 10.1177/10591478241255066.

Christopher Bayliss, Leandro do C. Martins, and Angel A. Juan. A two-phase local search with a discrete-event heuristic for the omnichannel vehicle routing problem. *Computers & Industrial Engineering*, 148:106695, 2020. doi: 10.1016/J.CIE.2020.106695.

Armagan Bayram and Bahriye Cesaret. Order fulfillment policies for ship-from-store implementation in omni-channel retailing. *European Journal of Operational Research*, 294 (3):987–1002, 2021. doi: 10.1016/J.EJOR.2020.01.011.

Radhika Bhargava, Rodrigo Reyes Levalle, and Shimon Y. Nof. A best-matching protocol for order fulfillment in re-configurable supply networks. *Computers in Industry*, 82: 160–169, 2016. doi: 10.1016/J.COMPIND.2016.07.001.

Bahriye Cesaret and Armagan Bayram. Analysis of customer switching behavior in omnichannel retailing. In *IIE Annual Conference. Proceedings*, pages 465–470. Institute of Industrial and Systems Engineers (IISE), 2020. URL https://www.proquest.com/docview/2522430228.

Guanting Chen, Xiaocheng Li, and Yinyu Ye. Technical note—an improved analysis of LP-based control for revenue management. *Operations Research*, 72(3):1124–1138, 2022. doi: 10.1287/opre.2022.2358.

Daniela Coppola. Retail e-commerce sales growth worldwide 2017-2026. Technical report, Statista, 2023. URL https://www.statista.com/statistics/288487/forecast-of-global-b2c-e-commerce-growth/.

Jyotirmoy Nirupam Das, Manoj Kumar Tiwari, Ashesh Kumar Sinha, and Vivek Khanzode. Integrated warehouse assignment and carton configuration optimization using deep clustering-based evolutionary algorithms. *Expert Systems with Applications*, 212: 118680, 2023a. doi: 10.1016/J.ESWA.2022.118680.

S. Das, R. Ravi, and S. Sridhar. Order fulfillment under pick failure in omnichannel ship-from-store programs. *Manufacturing & Service Operations Management*, 25(2):508–523, 2023b. doi: 10.1287/msom.2022.1164.

Qiyuan Deng, Xiaobo Li, Yun Fong Lim, and Fang Liu. Optimal policies and heuristics to match supply with demand for online retailing. *Manufacturing & Service Operations Management*, 26(5):1925–1944, 2024. doi: 10.1287/msom.2021.0394.

Levi DeValve. Cost balancing for general inventory/fulfillment networks with applications to ATO and multi-item e-retail problems. *SSRN Electronic Journal*, 2023. doi: 10.2139/ssrn.3961613.

Levi DeValve and Jabari Myles. Approximation algorithms for dynamic inventory management on networks. *Management Science*, 71(7):5893–5909, 2024. doi: 10.1287/mnsc.2022.02965.

Levi DeValve, Yehua Wei, Di Wu, and Rong Yuan. Understanding the value of fulfillment flexibility in an online retailing environment. *Manufacturing & Service Operations Management*, 25(2):391–408, 2021. doi: 10.1287/msom.2021.0981.

Banu Y. Ekren, Sara Perotti, Laura Foresti, and Lorenzo Prataviera. Enhancing e-grocery order fulfillment: improving product availability, cost, and emissions in last-mile delivery. *Electronic Commerce Research*, 25:2417–2455, 2025. doi: 10.1007/s10660-023-09799-x.

Awi Federgruen and Paul Zipkin. Allocation policies and cost approximations for multilocation inventory systems. *Naval Research Logistics Quarterly*, 31(1):97–129, 1984. doi: 10.1002/nav.3800310112.

Rong Gao, Yebao Ma, and Dan A. Ralescu. Uncertain multilevel programming with application to omni-channel vehicle routing problem. *Journal of Ambient Intelligence and Humanized Computing*, 1:1–13, 2022. doi: 10.1007/S12652-022-04419-2.

Zuguang Gao, Zi Ling, Varun Gupta, and Linwei Xin. Real-time omnichannel fulfillment optimization. *SSRN Electronic Journal*, 2023. doi: 10.2139/ssrn.4314250.

J. Goedhart, R. Haijema, R. Akkerman, and S. de Leeuw. Replenishment and fulfilment decisions for stores in an omni-channel retail network. *European Journal of Operational Research*, 311(3):1009–1022, 2023. doi: 10.1016/j.ejor.2023.06.018.

Aravind Govindarajan, Amitabh Sinha, and Joline Uichanco. Joint inventory and fulfillment decisions for omnichannel retail networks. *Naval Research Logistics*, 68(6):779–794, 2021. doi: 10.1002/NAV.21969.

Alexander Hübner, Andreas Holzapfel, and Heinrich Kuhn. Operations management in multi-channel retailing: an exploratory study. *Operations Management Research*, 8(3-4):84–100, 2015. doi: 10.1007/S12063-015-0101-9.

Alexander Hübner, Jonas Hense, and Christian Dethlefs. The revival of retail stores via omnichannel operations: A literature review and research framework. *European Journal of Operational Research*, 302(3):799–818, 2022. doi: 10.1016/J.EJOR.2021.12.021.

Stefanus Jasin and Amitabh Sinha. An LP-based correlated rounding scheme for multi-item ecommerce order fulfillment. *Operations Research*, 63(6):1336–1351, 2015. doi: 10.1287/OPRE.2015.1441.

Stefanus Jasin, Yanzhe (Murray) Lei, Sentao Miao, Debjit Roy, and Zhiyuan Sun. Joint inventory, pricing, and fulfillment control for a multi-warehouse multi-store problem: An asymptotically optimal lagrangian policy. *SSRN Electronic Journal*, 2022. doi: 10.2139/ssrn.4298794.

Song Jiu. Robust omnichannel retail operations with the implementation of ship-from-store. *Transportation Research Part E: Logistics and Transportation Review*, 157: 102550, 2022. doi: 10.1016/J.TRE.2021.102550.

Song Jiu, Dan Wang, and Zujun Ma. Benders decomposition for robust distribution network design and operations in online retailing. *European Journal of Operational Research*, 315(3):1069 – 1082, 2024. doi: 10.1016/j.ejor.2024.01.046.

Mehmet Karaoğlu and Gökhan Kara. Two-stage heuristic algorithm proposal for urban e-commerce deliveries. *Asia-Pacific Journal of Operational Research*, 39(03):2150034, 2022. doi: 10.1142/S0217595921500342.

Ruengsak Kawtummachai and Nguyen Van Hop. Order allocation in a multiple-supplier environment. *International Journal of Production Economics*, 93-94:231–238, 2005. doi: 10.1016/j.ijpe.2004.06.022. Proceedings of the Twelfth International Symposium on Inventories.

Ali Koc, Michael Desimone, Ajay Deshpande, Arun Hampapur, Yingjie Li, Xuan Liu, Christopher Milite, L. Brian Quanz, Raymond Tan, and Yada Zhu. An optimization framework for a multi-objective omni-channel order fulfilment problem. In *IIE Annual Conference. Proceedings*, pages 2051–2056. Institute of Industrial and Systems Engineers (IISE), 2017. URL https://research.ibm.com/publications/an-optimization-framework-for-a-multi-objective-omni-channel-order-fulfilment-problem.

Yanzhe (Murray) Lei, Stefanus Jasin, and Amitabh Sinha. Joint dynamic pricing and order fulfillment for e-commerce retailers. *Manufacturing & Service Operations Management*, 20(2):269–284, 2018. doi: 10.1287/MSOM.2017.0641.

Yanzhe Murray Lei, Stefanus Jasin, Joline Uichanco, and Andrew Vakhutinsky. Joint product framing (display, ranking, pricing) and order fulfillment under the multinomial logit model for e-commerce retailers. *Manufacturing & Service Operations Management*, 24 (3):1529–1546, 2021. doi: 10.1287/MSOM.2021.1012.

Shuqin Li and Shuai Jia. A benders decomposition algorithm for the order fulfilment problem of an e-tailer with a self-owned logistics system. *Transportation Research Part E: Logistics and Transportation Review*, 122:463–480, 2019. doi: 10.1016/J.TRE.2018.12.007.

Xiangyong Li, Jieqi Li, Y. P. Aneja, Zhaoxia Guo, and Peng Tian. Integrated order allocation and order routing problem for e-order fulfillment. *IISE Transactions*, 51(10): 1128–1150, 2019. doi: 10.1080/24725854.2018.1552820.

Yali Li and Jun Yang. The last-mile delivery vehicle routing problem with handling cost in the front warehouse mode. *Computers & Industrial Engineering*, 190:110076, 2024. doi: 10.1016/j.cie.2024.110076.

Ya'nan Li, Laijun Wang, and Liting Wang. Research on the urban logistics distribution routing problem based on the right of way. In *CICTP 2020: Advanced Transportation Technologies and Development-Enhancing Connections - Proceedings of the 20th COTA International Conference of Transportation Professionals*, pages 1872–1884. American Society of Civil Engineers, 2020. doi: 10.1061/9780784482933.161.

Yun Fong Lim, Song Jiu, and Marcus Ang. Integrating anticipative replenishment allocation with reactive fulfillment for online retailing using robust optimization. *Manufacturing & Service Operations Management*, 23(6):1616–1633, 2020. doi: 10.1287/MSOM.2020.0926.

Will Ma. Order-optimal correlated rounding for fulfilling multi-item e-commerce orders. *Manufacturing & Service Operations Management*, 25(4):1324–1337, 2023. doi: 10.1287/msom.2023.1219.

Stephen Mahar and P. Daniel Wright. The value of postponing online fulfillment decisions in multi-channel retail/e-tail organizations. *Computers & Operations Research*, 36(11):3061–3072, 2009. doi: 10.1016/J.COR.2009.02.007.

Stephen Mahar, Sachin Modi, and Peter A. Salzarulo. Appreciating how your bread is buttered: improving online order allocation for cross-channel retailers. *International Journal of Production Research*, 60(15):4845–4867, 2022. doi: 10.1080/00207543.2021.1939455.

Leandro Do C. Martins, Christopher Bayliss, Pedro J. Copado-Méndez, Javier Panadero, and Angel A. Juan. A simheuristic algorithm for solving the stochastic omnichannel vehicle routing problem with pick-up and delivery. *Algorithms*, 13(9):237, 2020a. doi: 10.3390/A13090237.

Leandro Do C. Martins, Christopher Bayliss, Angel A. Juan, Javier Panadero, and Mage Marmol. A savings-based heuristic for solving the omnichannel vehicle routing problem with pick-up and delivery. *Transportation Research Procedia*, 47:83–90, 2020b. doi: 10.1016/J.TRPRO.2020.03.082.

Leandro do C. Martins, Patrick Hirsch, and Angel A. Juan. Agile optimization of a two-echelon vehicle routing problem with pickup and delivery. *International Transactions in Operational Research*, 28(1):201–221, 2021. doi: 10.1111/ITOR.12796.

M. Mathirajan. Development of solution methodologies for online order fulfillment in omnichannel retailing. In *Proceedings of the International Conference on Industrial Engineering and Operations Management*, page 409. IEOM Society, 2021. URL https://www.researchgate.net/publication/375515272_Development_of_Solution_Methodologies_for_Online_Order_Fulfillment_in_Omnichannel_Retailing.

Marco Melacini, Sara Perotti, Monica Rasini, and Elena Tappia. E-fulfilment and distribution in omni-channel retailing: a systematic literature review. *International Journal of Physical Distribution and Logistics Management*, 48(4):391–414, 2018. doi: 10.1108/IJPDLM-02-2017-0101.

Ming Ni, Qing He, Xuan Liu, and Arun Hampapur. Same-day delivery with crowdshipping and store fulfillment in daily operations. *Transportation Research Procedia*, 38:894–913, 2019. doi: 10.1016/J.TRPRO.2019.05.046.

Bahareh Pourahmadi, Sadoullah Ebrahimnejad, and Misagh Rahbari. Scenario-based robust optimization for online retail orders considering supply chain costs and level of service: a real case study. *Sadhana - Academy Proceedings in Engineering Sciences*, 48 (4):240, 2023. doi: 10.1007/s12046-023-02282-1.

Warren B. Powell. *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions*. John Wiley & Sons, 2022. doi: 10.1002/9781119815068.

Meng Qi, Ho-Yin Mak, and Zuo-Jun Max Shen. Data-driven research in retail operations—a review. *Naval Research Logistics (NRL)*, 67(8):595–616, 2020. doi: 10.1002/nav.21949.

Louis Rivière, Christian Artigues, Azeddine Cheref, Nicolas Jozefowiez, Marie José Huguet, Sandra U. Ngueveu, and Vincent Charvillat. Multi-product, multi-supplier order assignment and routing for an e-commerce application in the retail sector. In *ICORES 2021 - Proceedings of the 10th International Conference on Operations Research and Enterprise Systems*, pages 438–445. SciTePress, 2021. doi: 10.5220/0010318304380445.

Ting Ting Shan and Xiao Ya Tian. Joint optimization on order splitting and order consolidation for online supermarket retailing. In *2022 41st Chinese Control Conference (CCC)*, volume 2022-July, pages 7532–7539. IEEE Computer Society, 2022. doi: 10.23919/CCC55666.2022.9901918.

Cong Shi, Yehua Wei, and Yuan Zhong. Process flexibility for multiperiod production systems. *Operations Research*, 67(5):1300–1320, 2019. doi: 10.1287/OPRE.2018.1810.

J. Tang, C. Qi, and H. Wang. Integrated optimization of order splitting and distribution routing for the front warehouse mode e-retailing. *International Journal of Production Research*, 2023. doi: 10.1080/00207543.2023.2200556.

Daniel Taylor, Sebastian Brockhaus, A. Michael Knemeyer, and Paul Murphy. Omnichannel fulfillment strategies: defining the concept and building an agenda for future inquiry. *International Journal of Logistics Management*, 30(3):863–891, 2019. doi: 10.1108/IJLM-09-2018-0223.

S. A. Torabi, E. Hassini, and M. Jeihoonian. Fulfillment source allocation, inventory transshipment, and customer order transfer in e-tailing. *Transportation Research Part E: Logistics and Transportation Review*, 79:128–144, 2015. doi: 10.1016/J.TRE.2015.04.004.

Hao Wang, Zhenzhen Yan, and Xiaohui Bei. A non-asymptotic analysis for re-solving heuristic in online matching. *Production and Operations Management*, 31(8):3096–3124, 2022a. doi: 10.1111/poms.13738.

Ke Wang, Yitian Li, and Yulin Zhou. Execution of omni-channel retailing based on a practical order fulfillment policy. *Journal of Theoretical and Applied Electronic Commerce Research*, 17(3):1185–1203, 2022b. doi: 10.3390/JTAER17030060.

Shixin Wang, Xuan Wang, and Jiawei Zhang. A review of flexible processes and operations. *Production and Operations Management*, 30(6):1804–1824, 2021. doi: 10.1111/POMS.13101.

Yihua Wang and Stefan Minner. Deep reinforcement learning for demand fulfillment in online retail. *International Journal of Production Economics*, 269:109133, 2024. doi: 10.1016/J.IJPE.2023.109133.

Lai Wei, Roman Kapuscinski, and Stefanus Jasin. Shipping consolidation across two warehouses with delivery deadline and expedited options for e-commerce and omni-channel retailers. *Manufacturing & Service Operations Management*, 23(6):1634–1650, 2020. doi: 10.1287/MSOM.2020.0903.

Ryan Wickman, Junxuan Li, and Xiaofei Zhang. VLS: A reinforcement learning-based value lookahead strategy for multi-product order fulfillment. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 14177:518 – 534, 2023. doi: 10.1007/978-3-031-46664-9_35.

Yongbo Xiao, Frank Y. Chen, and Jian Chen. Optimal inventory and dynamic admission policies for a retailer of seasonal products with affiliate programs and drop-shipping. *Naval Research Logistics*, 56(4):300–317, 2009. doi: 10.1002/NAV.20346.

Yaqi Xie, Will Ma, and Linwei Xin. The benefits of delay to online decision making. *Management Science*, 72(4):2826–2841, 2025. doi: 10.1287/mnsc.2023.00549.

Ping Josephine Xu, Russell Allgor, and Stephen C. Graves. Benefits of reevaluating real-time order fulfillment decisions. *Manufacturing & Service Operations Management*, 11 (2):340–355, 2009. doi: 10.1287/MSOM.1080.0222.

Zhen Xu, Hailun Zhang, Jiheng Zhang, and Rachel Q. Zhang. Online demand fulfillment under limited flexibility. *Management Science*, 66(10):4667–4685, 2020. doi: 10.1287/MNSC.2019.3449.

Yimo Yan, Andy H. F. Chow, Chin Pang Ho, Yong-Hong Kuo, Qihao Wu, and Chengshuo Ying. Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities. *Transportation Research Part E: Logistics and Transportation Review*, 162:102712, 2022. doi: 10.1016/j.tre.2022.102712.

J. Yang and Y. Li. A multicommodity pickup and delivery problem with time windows and handling time in the omni-channel last-mile delivery. *International Transactions in Operational Research*, 2023. doi: 10.1111/itor.13362.

Jingwei Yang, Qingchun Hou, Xiaoqing Wang, Yang Wei, Yuming Deng, Hongyang Jia, and Ning Zhang. Learning to solve an order fulfillment problem in milliseconds with edge-feature-embedded graph attention. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=qPQRIj_Y_EW.

Yanju Zhang, Liping Ou, and Jiaxu Liu. Multi-store collaborative delivery optimization based on top-k order-split. *PLOS ONE*, 18(3):e0278690, 2023. doi: 10.1371/journal.pone.0278690.

Yanyang Zhao, Xinshang Wang, and Linwei Xin. Multi-item online order fulfillment in a two-layer network. *Operations Research*, 73(5):2297–2305, 2025. doi: 10.1287/opre.2022.0100.

Yong Wu Zhou, Xiong Zhang, Yuanguang Zhong, Bin Cao, and T. C.Edwin Cheng. Dynamic pricing and cross-channel fulfillment for omnichannel retailing industry: An approximation policy and implications. *Transportation Research Part E: Logistics and Transportation Review*, 156:102524, 2021. doi: 10.1016/J.TRE.2021.102524.

Yada Zhu, Ajay Deshpande, L. Brian Quanz, L. Cao, Ali Koc, Xuan Liu, Yingjie Li, Arun Hampapur, Christopher Milite, Michael Desimone, and Raymond Tan. Improving e-commerce order fulfillment for peak times via incorporating fulfillment network load balancing. In *IIE Annual Conference. Proceedings*, pages 1806–1810. Institute of Industrial and Systems Engineers (IISE), 2017. URL https://research.ibm.com/publications/improving-e-commerce-order-fulfillment-for-peak-times-via-incorporating-fulfillment-network-load-balancing.

CHAPTER 3

Order Allocation in Online Marketplaces

Order Allocation in Online Marketplaces Considering Vendor Satisfaction

Sérgio Vasconcelos* · **Willem van Jaarsveld†** · **Gonçalo Figueira*** · **Bernardo Almada-Lobo***

Manuscript submitted to *European Journal of Operational Research*, being processed.

Abstract Online marketplaces depend on independent vendors to provide broad assortments and commit inventory. When the same item is offered by multiple vendors, the marketplace may determine which vendor fulfills its demand, therefore affecting the business each vendor receives and its incentives to sustain or expand its participation. The marketplace, therefore, has an interest, beyond operational efficiency, in how business for each item is distributed across vendors. This interest is captured in vendor-specific allocation targets, whose shortfall incurs an end-of-season penalty. This allocation process is difficult to manage because orders arrive sequentially, future demand is uncertain, and target attainment may conflict with low-cost fulfillment and customer service. To study this issue, we introduce the Multi-Item Order Fulfillment Problem in Online Marketplaces and formulate it as a Markov Decision Process.

We evaluate two myopic policies, two cost-function policies, a correlated-rounding policy, and deep controlled learning, a reinforcement-learning method combining approximate policy iteration with supervised learning. The computational study uses realistic instances motivated by a luxury fashion marketplace and includes a vendor-target stress test and structural scenario variants.

The results reveal a fulfillment-cost/target-shortfall trade-off whose severity depends on target pressure and marketplace structure. Higher target ambition increases shortfall but can reduce the proportion that policies can eliminate, while stronger penalties improve target steering only up to a saturation point. Deep controlled learning generally yields more cost-efficient trade-offs, whereas the primal-dual cost-function policy often achieves larger shortfall reductions at higher cost. Overall, non-myopic methods are most valuable when

*INESC TEC, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal

†Eindhoven University of Technology, Eindhoven, The Netherlands

current allocations affect future fulfillment opportunities.

Keywords E-commerce ·Online marketplaces ·Order allocation ·Vendor satisfaction ·Fulfillment optimization

3.1. Introduction

An online marketplace can be defined as an e-commerce site that brings sellers (vendors) and buyers (customers) together in one place, where all transactions are managed by the website owner. The importance of online marketplaces in the retail sector has rapidly increased, with the Gross Merchandise Value (GMV) of the top 100 global marketplaces doubling from 2018 to 2024. The most popular marketplaces have hundreds of millions of monthly visits with a strong presence in multiple markets, including the United States, Europe, Latin America, and China.

Alibaba operated the world's largest retail commerce business by GMV during the twelve months ended in March 31, 2026 (Alibaba Group Holding Limited, 2026). Reaching and sustaining this scale, however, requires marketplaces to build an attractive supply base, particularly during their early development, when demand and vendor trust are not yet self-sustaining. Jack Ma, Alibaba's co-founder, recounts that, when sellers first began offering products online but buyers were absent, Alibaba employees bought everything those sellers listed to demonstrate that the platform could generate transactions (Ma, 2015). This account illustrates the supply-side challenge faced by developing marketplaces, where vendors must see evidence that participation produces business before they are willing to broaden their assortment and commit more inventory.

The marketplace benefits from active vendor participation because vendors determine not only whether to join or remain on the platform, but also the breadth of the assortment and inventory they make available. Greater product variety and stock availability can attract customers within the vendors' segments (TechNet, 2024), while vendors' continued participation depends on whether the platform is perceived as useful and capable of converting participation into business value (Sun, 2010; Shen et al., 2022). Accordingly, marketplaces have an incentive to manage the sales opportunities received by vendors, especially when seeking to retain them or encourage broader assortment and inventory commitment.

One way to represent these objectives is through minimum allocation targets. Vendor-wide targets protect aggregate vendor performance but can be met through a small number of popular items. Item–vendor targets instead require allocations across the vendor's assortment and are therefore particularly appropriate when the marketplace seeks to motivate vendors to maintain broad offerings and commit inventory across their portfolios. Our work is thus motivated by item–vendor targets, while recognizing that vendor-wide targets remain suitable when aggregate vendor performance is the main concern.

From a fulfillment-optimization literature perspective, however, the vendor side has largely been ignored. Existing online order-allocation models primarily minimize fulfillment costs and unfulfillment, even though a marketplace that selects among vendors offering the same item can also use allocation decisions to influence the business each vendor

receives. We therefore study how to steer arriving demand toward vendors with unmet targets while preserving fulfillment efficiency.

This decision is challenging because orders must be allocated immediately, future demand is uncertain, inventory is finite, package-level fulfillment costs couple the units of an order, and target attainment may conflict with fulfillment and customer-service objectives. We introduce the Multi-Item Order Fulfillment Problem in Online Marketplaces (MIOFP-OM), in which unmet item–vendor or vendor-wide targets generate end-of-season penalties, and formulate it as a Markov Decision Process with Exogenous Inputs (MDP-EI). The computational study uses a reference scenario informed by a partnership with an established luxury fashion marketplace and structural variants representing alternative operational conditions.

Literature proposes various methods to optimize OAOR. The simplest ones are myopic methods, which generally correspond to allocating demand to the closest fulfillment node. These methods attempt to minimize the immediate fulfillment expense and/or the order delivery time. While simple, these methods completely ignore future demand, which can be significantly sub-optimal (see examples in Xu et al. (2009) and Acimovic and Graves (2015)). Other approaches resort to mathematical programming (MP). For example, Acimovic and Graves (2015) utilize linear programming to estimate the expected future costs and proposes a solution method that balances immediate and expected future fulfillment costs. These approaches improve the optimality gap of myopic methods at the cost of added models and computational complexity. More recently, approaches based on DRL have started to appear. For example, Wang and Minner (2024) formulate the problem as an MDP and utilize Proximal Policy Optimization to optimize a NN representing the fulfillment policy. These approaches are more difficult to implement than myopic methods due to the need for data and computational power to train the policies. These methods, however, have been developed primarily to optimize operational performance rather than vendor-facing allocation objectives. Moreover, although new approaches are commonly compared with myopic benchmarks, the literature does not provide a common comparison spanning the myopic, mathematical-programming-based, and learning-based policy classes considered here (Vasconcelos et al., 2026).

Given the previous motivation, the contributions of this work are as follows:

1. We introduce the MIOFP-OM, which, to the best of our knowledge, is the first dynamic OAOR formulation to incorporate vendor allocation targets and explicit penalties for unmet targets;

2. We formulate the problem as an MDP-EI that supports both item–vendor and vendor-wide targets while capturing finite vendor-owned inventory, package-level fulfillment costs, unfulfillment, and terminal target shortfalls;

3. We provide a common experimental comparison of myopic, cost-function, correlated-rounding, and learning-based fulfillment policies that have generally been studied separately;

4. We characterize the fulfillment-cost/vendor-satisfaction trade-off through a vendor-target stress test and structural scenario variants, identifying when target shortfall is

controllable and when non-myopic policies provide the greatest value.

The remainder of the paper is organized as follows. In Section 3.2, we review three streams of literature directly related to this work, Fulfillment Optimization, Data-driven methods and RL in inventory management, and Online marketplaces. In Section 3.3, we introduce the MIOFP-OM, state multiple assumptions and formalize the problem as a MDP-EI. In Section 3.4, we introduce the multiple methods adopted from online OF literature and Deep Controlled Learning (DCL). In Section 3.5, we describe the experimental design motivated by a luxury fashion marketplace, introduce the reference scenario and performance measures, and evaluate the fulfillment-cost/vendor-satisfaction trade-off through a vendor-target stress test. We then analyze structural scenario variants to assess how target attainability and policy performance change across alternative marketplace configurations. Section 3.6 concludes the paper.

3.2. Literature

Our work relates to three main topics, which we review next: fulfillment optimization, data-driven inventory management and optimization of online marketplace operations.

3.2.1 Fulfillment Optimization

A substantial literature studies online, multichannel, and omnichannel fulfillment optimization, but its objectives predominantly concern fulfillment cost, service, unfulfillment, and inventory utilization; the implications of allocation decisions for independent marketplace vendors remain largely unaddressed. One of the first major reviews on the topic by Agatz et al. (2008) identifies multiple issues in e-fulfillment optimization and groups them into two major topics, (1) supply management, which includes distribution network design, warehouse design, and inventory and capacity management, and (2) sales and delivery planning, which include delivery service design, pricing, forecasting, order promising, revenue management and transportation planning. Other more recent reviews have proposed similar topics and focus on specific contexts, such as multi-channel fulfillment (Hübner et al. (2015), focusing on integration with retail stores) and omnichannel fulfillment (Hübner et al. (2016), focusing on last-mile fulfillment, Melacini et al. (2018) focusing on distribution network design, inventory and capacity management and delivery planning and execution, and Hübner et al. (2022), focusing on retail store integration). Our work is more closely related to inventory management and delivery planning.

Acimovic and Farias (2019) identify a major issue in inventory management and delivery planning: in practice, inventory is often unbalanced with respect to the fulfillment objectives to be optimized (e.g., high amounts of inventory often accumulate close to low-demand regions whereas inventory is scarce close to high-demand regions, leading to high delivery costs). They identify four levers to manage this issue: (a) Transship inventory from one location to another, (b) Change the price of the item, (c) Increase the desirability of the product through marketing, and (d) Optimize the location from which to fulfill each order. Our work relates to the last lever.

Several works have contributed to the optimization of order allocation. The first to explore this issue was Xu et al. (2009), who proposed delaying fulfillment up to the ship-to-ready date (the time when OF needs to start to avoid delivery delays), and optimizing allocation for a batch of orders. However, as noted by Jasin and Sinha (2015), this is more and more difficult to do since delivery deadlines are becoming shorter and hence allocation immediately upon order arrival is more practical. Acimovic and Graves (2015) introduced a solution method to non-myopically solve the order allocation problem with allocations immediately upon order arrival, which consists of computing expected future fulfillment costs through a linear program and adding them to the immediate cost, showing significant improvements. Later, Jasin and Sinha (2015) focused on the multi-item problem by introducing a lookahead stochastic programming model whose solution is utilized in a complex randomized fulfillment scheme. Similar approaches to multi-item OF have been proposed by later works, such as Amil et al. (2025) and Ma (2023). Our work contributes to this literature by introducing vendor allocation targets and shortfall penalties as fulfillment objectives and by comparing several policy classes in a common experimental setting. For a comprehensive review on order allocation optimization, we refer the reader to (Vasconcelos et al., 2026).

The first application of DRL to online order allocation was by Wang and Minner (2024). In this work, the authors model the OF problem for a single item as a semi-MDP and solve it using PPO. We include DCL as the learning-based policy within a broader comparison of fulfillment-policy classes and apply it to a multi-item marketplace problem with vendor targets.

3.2.2 Data-driven methods in inventory management

Data-driven inventory research has primarily addressed replenishment, inventory placement, safety stocks, and inventory control rather than the real-time allocation of demand across independent marketplace vendors. Wang and Minner (2025) study a multi-location inventory placement model for online retailers to fulfill customer demands whose decisions include in which DC to place inventory, how to set base-stock levels for inventory-holding DC, and from which DC to fulfill customer demand. A data-driven stochastic program is proposed under two different demand fulfillment policies, fixed and virtual pooling. Svoboda and Minner (2026) propose a big-data solution for optimizing inventory buffers in large-scale supply networks. The authors optimize the placement and sizing of strategic inventories in multi-echelon supply chains by adapting the guaranteed-service model to introduce a nonparametric, big-data approach to supply chain safety-stock optimization based on mixed-integer linear programming.

There have been recent contributions to solving inventory management problems utilizing RL approaches. Boute et al. (2022) discuss the design choices associated with designing DRL algorithms to inventory control, and discuss potential avenues of future research including DRL for data-driven inventory control, blending analytical and numerical approaches and improving algorithmic scalability. De Moor et al. (2022) propose improving the performance of DRL in inventory control of perishable goods by introducing reward-shaping to a deep Q-network algorithm.

Accordingly, our use of DCL connects data-driven inventory control with online fulfillment decisions that incorporate vendor-side allocation objectives.

3.2.3 Online marketplaces and platform-mediated ranking

Marketplace operations research has examined business-model choice, seller participation, pricing, and platform-provided fulfillment services, but not how a marketplace should dynamically distribute incoming demand across vendors while accounting for vendor allocation objectives.

Li et al. (2024a) compare two common business models for an intermediary in a two-sided market, dealership, where the intermediary buys and resells products, and marketplace, where an intermediary provides a platform to connect the two sides, the seller and buyer. The authors find under which conditions, from the intermediary perspective, one model is preferred over the other, considering the pricing power, inventory risk, fulfillment times, and other factors. Tian et al. (2018) compare similar models, considering also the possibility of adopting a hybrid model, and find that the interaction of order-fulfillment costs and upstream competition influences the selection of an optimal model for the intermediary.

From the seller's perspective, Ryan et al. (2012) look at whether it should opt to sell its products in a marketplace or through its website, comparing the trade-off between expanding the reach of its business with the marketplace's revenue sharing scheme. Considering a seller operating in a marketplace, Sun et al. (2020) propose a generalized linear model that predicts its expected rewards by fulfilling a product themselves or utilizing fulfillment services provided by the marketplace, like Fulfillment By Amazon, to optimize the fulfillment channel of each product. Li et al. (2024b) look at how the competition between the sellers and marketplace influences the fulfillment services that the marketplace offers to the seller, finding that when competition does not exist, the provision of services is advantageous, improving the buyer's valuation of the vendor's products and the overall fulfillment efficiency, whereas when competition exists such benefits may not be observed.

These studies primarily address strategic or tactical decisions. In contrast, we study operational demand allocation and link platform-controlled fulfillment decisions to vendor participation, assortment breadth, and inventory commitment.

A related stream of work studies fairness in platform-mediated allocation and ranking decisions. Singh and Joachims (2018) propose a framework for fairness of exposure in rankings, motivated by settings where platforms must balance user utility with the exposure received by ranked items or providers. While their setting concerns exposure opportunities created through rankings, ours concerns realized fulfillment allocations subject to inventory availability and shipping costs; in both cases, platform decisions shape the opportunities received by participating agents.

3.3. Problem Setting and Model

In this section, we introduce the MIOFP-OM, a dynamic order allocation problem faced by online marketplaces that coordinate fulfillment across multiple vendor-owned stockpoints.

We describe the marketplace fulfillment setting and the role of vendor allocation targets, state the main modeling assumptions, and formulate the problem as an MDP-EI.

3.3.1 Marketplace Order Fulfillment with Vendor Targets

We consider an online marketplace that receives customer orders over a finite selling season and centrally determines which vendor fulfills each requested unit. Let $k \in \mathcal{K} := \{1, \dots, K\}$ denote a vendor, $i \in \{1, \dots, n\}$ an item, and $j \in \{1, \dots, J\}$ a customer delivery region. Each vendor owns its inventory, and multiple vendors may offer the same item through the marketplace.

We adopt the following assumptions:

1. Each vendor owns a single fulfillment node, and the marketplace does not operate its own fulfillment nodes;
2. The marketplace has full responsibility for selecting the fulfillment vendor or vendors for each order;
3. Inventory is committed before the selling season, with no replenishment during the season;
4. Each order must be allocated immediately upon arrival, before the next order is observed;
5. Orders from different customers cannot be consolidated for fulfillment;
6. For a given vendor–region pair, the package cost does not depend on item characteristics such as size, weight, or value;
7. Allocation targets are expressed in units, and each unit of target shortfall incurs the same penalty regardless of the item or vendor.

The first two assumptions establish a one-to-one correspondence between vendors and stockpoints and reflect marketplaces in which the platform, rather than the customer, selects the fulfillment source. Settings with multiple facilities per vendor can be represented by defining targets at the facility level or aggregating allocations across facilities. Platform-controlled allocation is also consistent with featured-offer mechanisms, under which one seller is presented as the default purchase option; in the Amazon context, Raval (2023) reports that approximately 80% of sales occur through this option. Assumptions 3 and 4 reflect contexts in which inventory is planned before the season and rapid delivery requires immediate allocation. Assumptions 5 and 6 exclude consolidation across customers and item-dependent package costs, while allowing multiple units of the same order assigned to one vendor to be consolidated into a single package. Assumption 7 reflects our focus on allocation volume as the operational measure of vendor performance. Using a common per-unit shortfall penalty keeps the target metric comparable across item–vendor pairs, while heterogeneous priorities can still be represented through the target quantities.

Before the season, vendor k commits an initial inventory b_{ik}^1 of item i. Total seasonal demand is modeled as

$$D \sim \text{Poisson}(\mu^D),$$

where μ^D is the expected total demand over the selling season.

Demand arrives as a sequence of orders $t = 1, \ldots, T$. Order t is characterized by its delivery region j^t, order size L^t, and item-composition vector $\boldsymbol{d}^t$, where d_i^t is the number of units of item i requested and $L^t = \sum_i d_i^t$.* The delivery region and order size are sampled according to

$$j^t \sim O^{\text{region}}, \qquad L^t \sim \text{Geom}(1/\mu^{\text{order}}),$$

where the geometric distribution has support $\{1, 2, \ldots\}$ and mean μ^{order}. Conditional on L^t, each unit is independently assigned to an item according to the item-demand distribution O^{item}, thereby determining $\boldsymbol{d}^t$.

Let $S^t = \sum_{\bar{t}=1}^{t-1} L^{\bar{t}}$ denote the cumulative demand observed before order t, with $S^1 = 0$. The season terminates after the first order whose completion causes cumulative demand to reach or exceed the sampled seasonal demand:

$$T = \inf\left\{t \geq 1 : S^{t+1} \geq D\right\}.$$

Each requested unit is assigned either to a real vendor or to an unfulfillment option, represented by a fictitious vendor $k = 0$ with infinite inventory. We define $\mathcal{K}_0 := \{0\} \cup \mathcal{K}$. Unfulfillment may result from insufficient inventory or from preserving inventory for potentially more valuable future demand.

Let x_{ik}^t denote the number of units of item i from order t assigned to vendor $k \in \mathcal{K}_0$. Let $y_k^t \in \{0, 1\}$ indicate whether real vendor k fulfills at least one unit of order t, and let c_{kj} denote the cost of sending a package from vendor k to region j. The fulfillment cost of order t is

$$\text{FC}^t = \sum_{k \in \mathcal{K}} y_k^t c_{kj^t}. \tag{3.1}$$

Because c_{kj^t} is incurred only once when vendor k is used, this expression captures consolidation within an order. Unfulfilled demand incurs a per-unit cost c^{unf}, giving

$$\text{UC}^t = c^{\text{unf}} \sum_i x_{i0}^t. \tag{3.2}$$

Let b_{ik}^t denote the inventory of item i available at vendor k before order t. The total seasonal allocation of item i to vendor k is $A_{ik} = b_{ik}^1 - b_{ik}^{T+1}$. For item–vendor targets, let $\theta_{ik} \leq b_{ik}^1$ denote the minimum desired allocation of item i to vendor k. A common penalty

*Throughout the paper, bold symbols denote vectors or matrices collecting the corresponding scalar quantities over their omitted indices.

p is applied to each unit of shortfall:

$$\mathrm{VC}_{ik}^{T+1,\text{item-vendor}} = p\max\{\theta_{ik} - A_{ik}, 0\}. \tag{3.3}$$

This piecewise-linear penalty is also compatible with linear-programming-based policies.

Alternatively, targets may be evaluated at the vendor level. Let $A_k = \sum_{i=1}^{n} A_{ik}$ denote the total allocation to vendor k, and let $\theta_k \leq \sum_{i=1}^{n} b_{ik}^1$ be its vendor-wide target. The corresponding penalty is

$$\mathrm{VC}_{k}^{T+1,\text{vendor-wide}} = p\max\{\theta_k - A_k, 0\}. \tag{3.4}$$

Item–vendor targets evaluate each assortment component separately, while vendor-wide targets provide greater flexibility by allowing allocations above target for one item to compensate for lower allocations of another. The penalty coefficient p is common across vendors and items. Heterogeneous vendor or item priorities are represented through the target quantities, while target-specific penalty coefficients are left outside the scope of this work.

The resulting problem, which we refer to as the MIOFP-OM, is to select a feasible vendor or the unfulfillment option for each arriving unit while minimizing expected fulfillment costs, unfulfillment costs, and terminal target-shortfall penalties under finite vendor-owned inventory. We next formulate this sequential decision problem as an MDP-EI.

3.3.2 MDP-EI Formulation

We formulate the MIOFP-OM as a finite-horizon MDP-EI (Temizöz et al., 2025) because demand realization is independent of allocation decisions, i.e., the policy affects inventory availability and costs, but not the sequence of customer orders.

Accordingly, we define the MIOFP-OM as the tuple

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{W}, \Xi, f, C, \alpha, \boldsymbol{s}_0 \rangle,$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{W}$ is the exogenous input space, Ξ is the distribution of the exogenous inputs, f is the transition function, C is the cost function, α is the discount factor, and $\boldsymbol{s}_0$ is the initial state. In this work, the selling season is finite and costs are undiscounted, so we set $\alpha = 1$. We now define each component of the tuple.

State space A state contains all information observed by the policy and required to make a unit-allocation decision. At any decision epoch within order t, let u denote the index of the unit currently being allocated. The state is

$$\boldsymbol{s}^{t,u} = \left(\boldsymbol{b}^t, S^t, j^t, \boldsymbol{d}^t, i^{t,u}, \boldsymbol{w}^{t,u}\right),$$

where $i^{t,u}$ is the item corresponding to the uth unit of the order, and $w_{ik}^{t,u}$ records the partial allocation of item i to vendor $k \in \mathcal{K}$, or to the unfulfillment option if $k = 0$, in the current order before allocating unit u.

The initial state before the first order is $\boldsymbol{s}_0 = (\boldsymbol{b}^1, 0, \varnothing, \varnothing, \varnothing, \mathbf{0})$. The first order information is generated by the first exogenous input.

Action space At each unit-allocation decision, the action is the vendor (real or fictitious) selected to fulfill the current unit. For a state $\boldsymbol{s}^{t,u}$, the feasible action set is

$$\mathcal{A}_{\boldsymbol{s}^{t,u}} = \{0\} \cup \left\{k \in \mathcal{K} : b^t_{i^{t,u}k} - w^{t,u}_{i^{t,u}k} > 0\right\}.$$

Thus, a real vendor is feasible if it has remaining inventory of the current item after accounting for units of the same order already assigned to it. The fictitious vendor is always feasible.

Exogenous inputs The exogenous inputs generate the demand process described in the previous subsection. At the beginning of the selling season, total seasonal demand D is sampled from Poisson(μ^D). At each order arrival t, the exogenous input is

$$\xi^t = (j^t, L^t, \boldsymbol{d}^t),$$

where $\xi^t \sim \Xi$ follows the order-generation process defined above: the delivery region, order size, and item composition are sampled independently from their corresponding distributions.

Transition function The transition function f updates the state after each unit-allocation action. If action $k^{t,u} \in \mathcal{A}_{\boldsymbol{s}^{t,u}}$ is selected for the current unit $i^{t,u}$, the partial allocation matrix is updated as

$$w^{t,u+1}_{i^{t,u}k^{t,u}} = w^{t,u}_{i^{t,u}k^{t,u}} + 1,$$

with all other entries unchanged. If the order still contains unallocated units, the next state corresponds to the next unit of the same order.

After the last unit of order t is allocated, the partial allocation matrix becomes the complete order allocation $\boldsymbol{x}^t = \boldsymbol{w}^{t,L^t+1}$. Inventory and cumulative demand are then updated according to

$$b^{t+1}_{ik} = b^t_{ik} - x^t_{ik}, \qquad i = 1, \ldots, n,\ k \in \mathcal{K},$$

and

$$S^{t+1} = S^t + L^t.$$

If the selling season has not ended ($S^t < D$), a new order exogenous input ξ^{t+1} is sampled and the system moves to the first unit-allocation state of order $t+1$. Otherwise, the system transitions to a terminal state and the vendor-target penalty is incurred.

Cost function The cost function contains immediate order-related costs and a terminal vendor-target cost. Once order t is fully allocated, the immediate cost of order t is computed as

$$C^t = \mathrm{FC}^t + \mathrm{UC}^t .$$

For item–vendor targets, the terminal vendor-target cost is

$$C^{T+1} = \sum_{ik} \mathrm{VC}_{ik}^{T+1,\text{item–vendor}} .$$

Similarly, for vendor-wide targets, the terminal cost is

$$C^{T+1} = \sum_{k} \mathrm{VC}_{k}^{T+1,\text{vendor-wide}} .$$

The total trajectory cost is the sum of order costs and the terminal vendor-target cost.

Policies A policy specifies, for every state $\boldsymbol{s}$, a probability distribution over feasible actions,

$$\pi(\cdot \mid \boldsymbol{s}) \in \Delta(\mathcal{A}_{\boldsymbol{s}}),$$

where $\pi(a \mid \boldsymbol{s})$ is the probability of selecting action a in state $\boldsymbol{s}$, and $\Delta(\mathcal{A}_{\boldsymbol{s}})$ denotes the set of probability distributions supported on $\mathcal{A}_{\boldsymbol{s}}$. A policy is deterministic if it assigns probability one to a single action.

All policies considered in this work are deterministic except Dilate Round policy (R-D). At the beginning of each order, R-D samples vendor opening times and reuses these random draws when allocating the units of that order, inducing correlated assignments within the order. Conditional on the sampled opening times, its unit-allocation decisions are deterministic.

Because decisions are made at the unit level, the complete allocation of an order is obtained by repeatedly applying the policy to the sequence of unit-allocation states induced by that order. The objective is to find a policy that minimizes the expected total cost over the selling season:

$$\min_{\pi} \mathbb{E}^{\pi}\left[\sum_{t=1}^{T} C^t + C^{T+1}\right],$$

where the expectation is taken over the exogenous demand process and, for R-D, the policy's auxiliary random variables.

Algorithms 1 and 2, presented in Appendix 3.A, provide a procedural view of the same dynamics and clarify the nesting between order arrivals and unit-allocation decisions. Algorithm 1 describes the selling-season evolution under a fixed policy, while Algorithm 2 shows how an order-level allocation is constructed through unit-level decisions. The same

procedural logic is used to implement the MDP-EI using the DynaPlex software library (Akkerman et al., 2023) for the computational experiments.

3.4. Methods

In this section, we describe the solution policies evaluated for the MIOFP-OM. The policy set includes two myopic policies: a cost-oriented policy that ignores vendor targets, Myopic Multi-Item policy (M-MI), and fulfillment cost and vendor target-aware policy, Myopic Satisfaction policy (M-S). These policies function as simple baselines (they do not require any training or tuning, but ignore the impact of decisions in the future), and their comparison allows studying the impact of satisfaction-aware myopic steering in Section 3.5.5.

The policy set also includes a set of forward-looking policies. It includes two policies that attempt to estimate future costs via a cost-function approximation: Linear-Programming-Based Cost Function policy (CF-LP) adapted from Acimovic and Graves (2015) and Primal-Dual-Based Cost Function policy (CF-PD), adapted from Andrews et al. (2019); and a randomized fulfillment policy with good asymptotic properties in multi-item order fulfillment, Dilate Round policy (R-D), adapted from Ma (2023). Finally, we apply DCL (Temizöz et al., 2025), a general purpose DRL policy designed for problems formulated as MDP-EI, which has not previously been used for online OF. This non-myopic policy set is intentionally diverse and enables a broad comparison of solution approaches in Sections 3.5.5 and 3.5.6.

We make two important remarks regarding these forward-looking policies. First, all except DCL were adapted to the MIOFP-OM from other order fulfillment formulations. The key change lies in including vendor targets and penalties in the policy design, via (1) adding new terms to the objective function and new constraints to the policy's underlying Deterministic Linear Program (DLP) (CF-LP and R-D) or (2) adapting the reward signal (CF-PD). Since some assumptions between our model and the other papers' were conflicting, other adaptations were required to adapt these policies. We forward the reader to the remainder of this section for more details.

Second, the policies differ in both their offline preparation requirements and their online execution effort. The M-MI and M-S policies require no additional policy-specific preparation and make greedy allocation decisions online. The R-D and DCL policies shift most of their computational effort offline. While R-D uses order-composition demand forecasts to solve a DLP once and reuses its solution through correlated rounding and repair, DCL trains a policy through simulation and subsequently evaluates the resulting NN online. The cost-function policies require different forms of preparation and online computation. CF-LP uses item-level demand forecasts and approximations of multi-item fulfillment savings to construct a DLP that is solved online. CF-PD instead requires its primal-dual parameters to be calibrated from historical data or simulation and uses them to update an approximate cost-function correction during allocation. These requirements are summarized in Table 3.1 and detailed in Appendix 3.F.

Table 3.1: Offline preparation and online execution requirements by policy.

Policy	Offline preparation	Online execution
M-MI	No additional policy-specific preparation.	Greedy allocation based on immediate fulfillment and unfulfillment costs.
M-S	No additional policy-specific preparation.	Greedy allocation based on satisfaction-adjusted immediate costs.
CF-LP	Estimate item-level demand and approximate multi-item fulfillment savings.	Solve the DLP and evaluate its inventory cost corrections during allocation.
CF-PD	Calibrate primal-dual cost-function parameters using historical data or simulation.	Update and evaluate the approximate cost-function correction during allocation.
R-D	Estimate order-composition demand and solve the DLP once.	Apply the correlated-rounding and repair procedure using the offline solution.
DCL	Train the policy through an iterative simulation-based procedure.	Evaluate the learned NN policy.

3.4.1 Myopic policies

Let $k^{\pi}_{t,u}$ denote the action selected by policy π in state $\boldsymbol{s}^{t,u}$, and let $i^{t,u}$ be the item currently being allocated.

For any feasible real vendor $k \in \mathcal{K}$, define the incremental package indicator

$$\Delta y^{t,u}_k = \begin{cases} 0, & \text{if } \sum_i w^{t,u}_{ik} > 0, \\ 1, & \text{otherwise.} \end{cases}$$

Thus, assigning the current unit to vendor k incurs the vendor–region shipping cost only if vendor k has not yet been used for the current order. We define the immediate cost of action $k \in \mathcal{A}_{\boldsymbol{s}^{t,u}}$ as

$$g^{t,u}_k = \begin{cases} \Delta y^{t,u}_k c_{kj^t}, & k \in \mathcal{K}, \\ c^{unf}, & k = 0. \end{cases}$$

M-MI ignores vendor satisfaction and selects the action with the lowest immediate cost:

$$k^{M-MI}_{t,u} = \arg\min_{k \in \mathcal{A}_{\boldsymbol{s}^{t,u}}} g^{t,u}_k.$$

Contrarily, M-S accounts for vendor satisfaction by subtracting the penalty that would be avoided if assigning the current unit to vendor k reduces a still-unmet target. For item–

vendor targets, define

$$\delta_k^{t,u} = \begin{cases} 1, & \text{if } k \in \mathcal{K} \text{ and } A_{i^{t,u}k}^{t,u} < \theta_{i^{t,u}k}, \\ 0, & \text{otherwise,} \end{cases}$$

where $A_{ik}^{t,u} = b_{ik}^1 - b_{ik}^t + w_{ik}^{t,u}$ denotes the number of units of item i already allocated to vendor k before allocating unit u of order t. The policy selects

$$k_{t,u}^{\text{M-S}} = \underset{k \in \mathcal{A}_{s^{t,u}}}{\arg\min} \left(g_k^{t,u} - p\,\delta_k^{t,u} \right).$$

For vendor-wide targets, the same expression is used, but $\delta_k^{t,u}$ is defined using the aggregate vendor allocation:

$$\delta_k^{t,u} = \begin{cases} 1, & \text{if } k \in \mathcal{K} \text{ and } A_k^{t,u} < \theta_k, \\ 0, & \text{otherwise,} \end{cases}$$

M-S can be interpreted as a one-step penalty-avoidance heuristic: it treats an allocation to a vendor with an unmet target as immediately reducing the terminal penalty by p, while ignoring the effect of the current allocation on future fulfillment opportunities.

3.4.2 Cost-function policies

These policies approximate the cost-to-go function by assigning an opportunity cost to the consumption of inventory. This follows the approximate-dynamic-programming interpretation commonly used in fulfillment optimization and revenue management: instead of optimizing only the immediate cost of assigning the current unit, the policy adjusts this cost by an estimate of the future value lost when one unit of inventory is depleted. We consider two policies that follow this logic, but estimate opportunity costs differently.

The first policy, CF-LP, is based on the linear-programming approach proposed by Acimovic and Graves (2015). In this approach, a DLP is solved to approximate the future fulfillment problem, and the dual values of the inventory constraints are used as estimates of the marginal value of inventory. This requires explicit information about expected remaining demand. In our implementation, the policy solves a live DLP for the item currently being allocated. The linear program approximates future demand for that item by customer region and distinguishes between single-unit demand, multi-unit demand that can be fulfilled without splitting the order, and multi-unit demand that may require splitting the order. The detailed formulation of this linear program, denoted by DLP^{CF-LP}, is provided in Appendix 3.B.1.

Let $\nu_{i^{t,u}k}^{t,u}$ denote the dual value of the inventory constraint for vendor k in the DLP^{CF-LP} solved for the current item $i^{t,u}$ in state $s^{t,u}$. Using the immediate cost $g_k^{t,u}$ and one-step penalty-avoidance term $p\,\delta_k^{t,u}$ defined in the myopic-policy subsection, CF-LP selects

$$k_{t,u}^{\text{CF-LP}} = \underset{k \in \mathcal{A}_{s^{t,u}}}{\arg\min} \left(g_k^{t,u} - p\,\delta_k^{t,u} - \nu_{i^{t,u}k}^{t,u} \right).$$

Thus, a vendor with a high opportunity cost becomes less attractive, because allocating the current unit to that vendor consumes inventory that is expected to be valuable for future demand. In the current implementation, CF-LP is used only with item–vendor targets, because the live DLP is constructed separately for the item currently being allocated.

The second policy, CF-PD, is based on the primal-dual approach of Andrews et al. (2019). Contrary to CF-LP, opportunity-cost estimates are represented via item–vendor multipliers that are updated online as inventory is allocated. This makes the policy less dependent on explicit demand forecasts, consistent with the motivation of primal-dual fulfillment approaches for settings in which granular demand forecasts are difficult to obtain.

Let $\lambda^{t,u}_{i^{t,u}k}$ denote the primal-dual multiplier associated with the inventory of the current item at vendor k. Let $R^{t,u}_k$ denote a reward-like transformation of the satisfaction-adjusted immediate cost $g^{t,u}_k - p\delta^{t,u}_k$, constructed so that lower adjusted costs produce larger rewards. The multiplier increases as inventory of the corresponding item–vendor pair is consumed. The policy selects

$$k^{\text{CF-PD}}_{t,u} = \arg\max_{k \in \mathcal{F}^{t,u}} \left(R^{t,u}_k - \lambda^{t,u}_{i^{t,u}k} \right).$$

If no real vendor has inventory for the current item, the unit is assigned to the unfulfillment action. After selecting vendor $k^{CF-PD}_{t,u}$, the corresponding multiplier is updated according to the primal-dual rule described in Appendix 3.B.2. The same appendix also reports the parameter-calibration procedure used for this policy[†].

3.4.3 Dilate rounding policy

Rounding policies are based on the randomized-fulfillment approach introduced in Jasin and Sinha (2015) and further developed in Ma (2023). Like CF-LP, they use a DLP to incorporate information about future demand. However, the role of the linear program is different. In CF-LP, a live DLP is solved during execution to estimate marginal inventory opportunity costs. In rounding policies, a DLP is solved offline before real-time execution. Its solution defines target fulfillment probabilities that are then used by a randomized online policy.

Let DLP^{R} denote the offline DLP used by the rounding policies. For each order type q, delivery region j, item $i \in q$, and vendor k, the DLP determines a probability r^q_{kij} with which item i in an order of type q from region j should be fulfilled by vendor k. If each item in a multi-item order were assigned independently according to the DLP probabilities r^q_{kij}, the long-run inventory flows would follow the DLP solution, but orders would tend to be split across too many vendors. The purpose of correlated rounding is to preserve the marginal assignment probabilities while inducing positive correlation across item assignments within the same order. This increases the probability that multiple items in an order are assigned to the same vendor, reducing order-level fulfillment costs.

We adapt the correlated rounding policy with dilated opening times proposed by Ma (2023), which we denote by R-D. The policy starts from the marginal assignment probabil-

[†]Although Andrews et al. (2019) propose specific parameter values, these values performed poorly in our setting; we therefore calibrate the parameters using simulation.

ities obtained from the offline DLP. Each vendor draws a random opening time, and each item observes these opening times on an item-specific dilated time scale chosen so that the resulting assignment probabilities match the DLP marginals. The item is then assigned to the vendor whose dilated opening time is observed first. We also adapted the force-open rounding policy proposed in Ma (2023), which extends the same correlated-rounding logic to include forced openings. In our experiments, however, its performance was very close to that of R-D, with R-D performing slightly better more often. We therefore focus on R-D in this paper.

Contrary to Jasin and Sinha (2015) and Ma (2023), our model does not impose any limit on the number of order types q that exist. Consequently, in our work, we only consider a restricted number of order types for the purpose of constructing DLP^{R}, and, consequently, the randomized fulfillment can only be applied to orders that are included in this set. If an arriving order type falls outside this set, R-D falls back to M-S.

In addition, because the DLP probabilities are computed offline, the rounded vendor may no longer have inventory for the current item when the order arrives. We therefore considered two repair variants for infeasible rounded assignments. The first repairs the assignment by applying M-S. The second assigns the unit to the unfulfillment action, as in Ma (2023). In our experiments, the unfulfillment-repair variant consistently performed worse because it generated excessive unfulfillment costs. Therefore, all R-D results reported in this paper use the myopic-repair variant.

The offline DLP formulation, correlated rounding procedure, and experimental parameterization, including the criteria for order type inclusion in DLP^{R}, are provided in Appendix 3.C.

3.4.4 DCL

DCL is a RL algorithm proposed by Temizöz et al. (2025) for inventory-control problems modeled as MDP-EIs. DCL is explicitly designed for settings in which uncertainty is driven by exogenous inputs, such as demand, and where evaluating an action from a single trajectory can be highly noisy.

DCL is based on approximate policy iteration. Starting from an initial policy π_0, the algorithm repeatedly constructs an improved policy by sampling states, estimating the best action for each sampled state through simulation, and training a NN classifier to imitate these simulation-based actions. Thus, DCL uses the NN only as a policy representation. The action labels used for training are obtained by comparing feasible actions through rollout simulations.

For a sampled state $\boldsymbol{s}$, DCL estimates the simulation-based improved action by evaluating each feasible action $a \in \mathcal{A}_{\boldsymbol{s}}$. After taking action a, the rollout follows the current policy π_g for a finite horizon of length H. To reduce the variance of action comparisons, the same exogenous scenarios are used when evaluating competing actions, following a common-random-numbers scheme. In addition, DCL uses sequential halving to allocate more simulation effort to actions that appear more promising.

The resulting dataset at approximate policy-iteration step g is

$$\mathcal{D}_g = \left\{(\boldsymbol{s}_m, \hat{\pi}_g^+(\boldsymbol{s}_m)) : m = 1, \ldots, N\right\},$$

where $\hat{\pi}_g^+(\boldsymbol{s}_m)$ is the simulation-based improved action for state $\boldsymbol{s}_m$. A NN classifier N_ψ is then trained on this dataset to obtain the next policy π_{g+1}. The learned policy maps states directly to allocation actions and is used as the base policy in the next approximate policy-iteration step.

In this work, we adapt DCL to the MDP-EI formulation in Section 3.3.2. Each action corresponds to selecting one of the feasible vendors for the current unit, or selecting the unfulfillment action. The initial policy is M-S, which provides a simple satisfaction-aware starting point. The rollout simulation uses the same MDP-EI dynamics described in Section 3.3.2, including fulfillment costs, unfulfillment costs, inventory transitions, and the terminal vendor-target penalty.

The NN policy receives a feature representation of the current unit-allocation state. The features include the expected remaining demand by item, the delivery region of the current order, remaining inventory for all items across all vendors, current vendor-specific immediate allocation costs adjusted for target relief, the remaining target requirement relevant to the current item for each vendor, the identity of the item currently being allocated, and the composition of the remaining units in the order. The full DCL implementation, including the sampling procedure, action-label generation, state-feature construction, neural-network architecture, and training parameters, is reported in Appendix 3.D.

3.5. Computational Experiments

We evaluate the policies introduced in Section 3.4 through a set of computational experiments designed to study the fulfillment-cost/vendor-target trade-off in marketplace order allocation. The section has two goals. First, it explains how the experimental environment is constructed, including the practical motivation, the instance-generation procedure, and the reference scenario used as the baseline setting. Second, it uses this environment to compare the proposed policies under different vendor-target requirements and operational conditions.

3.5.1 Experimental motivation

The experimental design is motivated by practical experience from collaborations with an online luxury apparel marketplace operating globally. This setting has several characteristics that make the MIOFP-OM relevant.

First, vendors often overlap in their product offerings, so the same customer demand can be fulfilled by multiple vendors. Second, customers do not choose which vendor fulfills each item; the marketplace centrally determines all allocation decisions. Third, vendor relationship management is an important operational concern, especially given the existence of contractual obligations with key vendors. In practice, the marketplace used a priority-based system to select vendors, but such systems can be too rigid: they may satisfy

vendor-prioritization rules while overlooking other fulfillment objectives, such as reducing shipping costs. The MIOFP-OM captures this trade-off explicitly: vendor prioritization is represented through allocation targets and penalties for target shortfalls, while fulfillment costs and unfulfillment costs remain part of the objective.

3.5.2 Instance generation

Each computational instance instantiates the components of the MDP-EI defined in Section 3.3.2. The construction of the instances uses industry-informed assumptions, public population and parcel-delivery data, and parameter choices motivated by the fulfillment literature. The instance-generation procedure combines fixed construction rules with configurable experimental dimensions. The fixed construction rules define how network costs, regional demand, order sizes, inventory quantities, and target quantities are computed once the relevant scenario parameters have been selected. The configurable dimensions determine which network, demand, inventory, and target settings are used in each experiment. The details on the instance-generation procedure, including the distinction between fixed rules and configurable dimensions, are described in Appendix 3.E.

The reference scenario in Section 3.5.3 selects one baseline configuration from these dimensions. The vendor-target stress test then varies the target ambition and penalty dimensions while keeping the target granularity, reference network, demand, and inventory configuration fixed. The scenario-variant experiment instead modifies selected scenario-configuration dimensions.

3.5.3 Reference scenario

The reference scenario defines the baseline operational environment used throughout the computational study. It is calibrated to represent a medium-sized fulfillment sub-network within a larger online marketplace. This is motivated by the fact that, although marketplaces may contain thousands of products and hundreds of vendors, direct competition is often concentrated within smaller product categories, where a limited number of vendors overlap on a limited set of items. We therefore set $K = 5$ vendors and $n = 20$ items, which creates a non-trivial allocation problem while remaining representative of a localized competitive sub-network.

The geographical and cost structure is designed to reflect a European marketplace setting, as detailed in Appendix 3.E.1. We define $J = 30$ delivery regions, corresponding to the countries covered by the parcel-cost dataset considered, and assume that customer-region demand is proportional to population. Vendors are located in distinct European countries, generating heterogeneous domestic and cross-border fulfillment costs.

Demand is generated to reflect the low-volume and skewed nature of luxury fashion sales. We consider a selling season of 10 weeks, corresponding approximately to a quarter-year apparel collection. Empirically, items sold by multiple vendors generate on average approximately 1.5 units of weekly demand, corresponding to 15 units over a 10-week season. This magnitude is consistent with Andrews et al. (2019), who report demand levels

implying approximately 2.9 units per item per fulfillment node over a season.[‡] Accordingly, the reference scenario sets average expected demand to $\bar{\mu} = 15$ units per item. Item-demand probabilities follow an ABC-like profile, so that a small number of items account for a larger share of expected demand while many items have lower expected demand. Orders are also small on average; hence, order sizes are sampled from a geometric distribution with mean $5/3$, reflecting that fashion orders typically contain few units.

Initial inventory is positioned in proportion to the demand each vendor is expected to serve under cost-efficient fulfillment. This level of alignment is unlikely to arise naturally in a decentralized marketplace, where vendors independently decide how much inventory to commit to the platform. We nevertheless use the cost-aligned option in the reference scenario to create a controlled baseline for the vendor-target stress test. Since inventory is already aligned with the fulfillment-cost structure, immediate-cost myopic policies operate under comparatively favorable conditions, allowing the experiment to focus more directly on the trade-off induced by vendor-target incentives.

Furthermore, we consider that vendor penalties are evaluated at the item–vendor level in the reference scenario. This level of granularity, rather than vendor-wide targets, follows the motivation in Section 3.1: promoting the sale of the entire vendor assortment motivates vendors to expand their assortments and stock depth, leading to more sustained marketplace growth. We also set the per-unit unfulfillment cost to twice the largest fulfillment route cost, $c^{\text{unf}} = 2\max_{k,j} c_{kj}$, so that rejecting demand is significantly more costly than any physical fulfillment option.

The reference scenario, therefore, fixes a realistic but relatively structured operational environment: demand is skewed, order sizes are small, fulfillment costs are geographically heterogeneous, inventory is initially aligned with cost-efficient regional assignment, and targets promote allocations across the entire vendor's portfolio. The subsequent experiments vary this baseline in two complementary ways. First, the vendor-target stress test keeps most of the environment fixed and varies only the target ambition and penalty intensity. Second, the scenario-variant experiment fixes a representative target ambition and penalty intensity and varies selected operational dimensions of the reference scenario. Table 3.2 summarizes this experimental structure.

3.5.4 Performance measures

To compare configurations and policies consistently, we next define the demand, target-shortfall, and fulfillment-cost measures used throughout the analysis. Within each configuration, all policies are evaluated on a common set Ω of 1,000 trajectories.

Let $Q^\omega = \sum_{t=1}^{T^\omega} L^{t,\omega}$ denote the number of demand units processed in trajectory ω, and let $\overline{Q} = \frac{1}{|\Omega|}\sum_{\omega\in\Omega} Q^\omega$ denote average realized demand. All vendor-target stress-test configurations and all structural variants except "More demand" and "Larger network" use expected seasonal demand $\mu^D = 300$. The latter two use $\mu^D = 600$. Across the evaluation trajectories, realized mean demand closely matches these values: 299.9 units in the former configurations, 599.9 units under "More demand", and 600.6 units in the larger network.

[‡] Andrews et al. (2019) report 50,617 total online orders over a season, across 129 items and 134 fulfillment nodes.

Table 3.2: Experimental configurations relative to the reference scenario. Note that the scenario variants modify one operational dimension at a time relative to the reference scenario.

Section	Scenario	Operational configuration	Target ambition and vendor-target penalty
Sections 3.5.5	Reference / stress test	Baseline operational configuration: item–vendor target granularity, $K = 5$, $n = 20$, ABC item-demand option, $\bar{\mu} = 15$, and cost-aligned inventory.	Stress test varies target ambition τ, and vendor-target penalty intensity β.
Section 3.5.6	Vendor-wide targets	Replaces item–vendor targets with aggregate vendor-wide targets.	$\tau = 0.50$ and $\beta = 0.25$.
	Symmetric inventory	Replaces cost-aligned inventory with symmetric inventory across vendors.	
	Sparse availability	Replaces full item–vendor overlap with sparse item–vendor availability.	
	Uniform demand	Replaces ABC item demand with equal expected demand across items.	
	Larger network	Doubles the number of vendors and items: $K = 10$, $n = 40$. Average expected demand per item is unchanged.	
	More demand	Doubles average expected demand per item: $\bar{\mu} = 30$, keeping $K = 5$ and $n = 20$.	

Let $\mathcal{R}$ contain the item–vendor pairs under item–vendor targets and the vendors under vendor-wide targets. For requirement $r \in \mathcal{R}$, let θ_r denote its target and $A_r^{\omega,\pi}$ the corresponding seasonal allocation under policy π. The total allocation target is $\Theta = \sum_{r\in\mathcal{R}} \theta_r$, and its ratio to realized demand is

$$\mathrm{TDR} = 100\frac{\Theta}{\overline{Q}}.$$

The target-to-demand ratio (TDR) describes aggregate target pressure. A value below 100% means that average realized demand exceeds the total allocation target, but does not guarantee that every granular target can be met. A value above 100% indicates that full attainment requires an above-average realization of demand.

The mean remaining target shortfall, in units and as a percentage of the total allocation target, is

$$\mathrm{SF}^{u}(\pi) = \frac{1}{|\Omega|}\sum_{\omega\in\Omega}\sum_{r\in\mathcal{R}}\left[\theta_r - A_r^{\omega,\pi}\right]^{+}, \qquad \mathrm{SF}^{\%}(\pi) = 100\frac{\mathrm{SF}^{u}(\pi)}{\Theta}. \tag{3.5}$$

All policy-relative measures use M-MI as the reference. We define shortfall reduction and fulfillment-cost delta as

$$\mathrm{SR}^{u}(\pi) = \mathrm{SF}^{u}(\text{M-MI}) - \mathrm{SF}^{u}(\pi), \qquad \mathrm{SR}^{\%}(\pi) = 100\frac{\mathrm{SR}^{u}(\pi)}{\mathrm{SF}^{u}(\text{M-MI})}, \tag{3.6}$$

$$\Delta\mathrm{FC}^{\%}(\pi) = 100\frac{\mathrm{FC}(\pi) - \mathrm{FC}(\text{M-MI})}{\mathrm{FC}(\text{M-MI})}, \tag{3.7}$$

where $\mathrm{FC}(\pi)$ is the mean seasonal fulfillment cost obtained by summing Equation (3.1) over the orders in each trajectory. Positive $\mathrm{SR}^{\%}$ indicates lower remaining shortfall than M-MI, whereas positive $\Delta\mathrm{FC}^{\%}$ indicates higher fulfillment cost.

Because policies may select the unfulfillment action, we also monitor mean unfulfilled demand to verify that fulfillment-cost improvements are not obtained by rejecting additional demand. However, we found that differences in mean unfulfilled demand between M-MI and other policies were negligible across the reported comparisons (the maximum was an average increase of 0.9 units of unfulfilled demand, out of 600 average units of demand, for DCL in the "More demand" scenario) and therefore do not affect the fulfillment-cost/shortfall trade-offs.

Table 3.3 summarizes the target scale and M-MI shortfall underlying both experiments. Section 3.5.5 varies the target ambition in the reference operating environment. Section 3.5.6 fixes $\tau = 0.50$ and $\beta = 0.25$ and compares the structural scenarios introduced in Table 3.2.

3.5.5 Vendor-target stress test

This experiment examines how target ambition and penalty intensity affect target pressure, shortfall controllability, and the fulfillment-cost consequences of vendor steering. Item–

Table 3.3: Target scale and M-MI remaining shortfall across the experimental configurations.

Configuration	Target-to-demand ratio, TDR	M-MI shortfall $SF^{\%}$ [SF^{u}]
Section 3.5.5		
$\tau = 0.25$	49.7%	16.44% [24.5]
$\tau = 0.50$ (Section 3.5.6 reference scenario)	77.7%	15.57% [36.3]
$\tau = 0.75$	120.7%	25.81% [93.4]
$\tau = 0.90$	137.1%	28.81% [118.4]
Section 3.5.6		
Vendor-wide targets	77.7%	1.75% [4.1]
Symmetric inventory	76.7%	15.61% [35.9]
Sparse item–vendor availability	67.0%	9.30% [18.7]
Uniform demand	86.7%	16.75% [43.6]
More demand	66.5%	7.47% [29.8]
Larger network	104.2%	33.22% [207.9]

vendor targets and the per-unit shortfall penalty are generated as

$$\theta_{ik} = \left\lceil \tau b^{1}_{ik} \right\rceil, \qquad p = \beta \max_{k,j} c_{kj}, \tag{3.8}$$

with $\tau \in \{0.25, 0.50, 0.75, 0.90\}$ and $\beta \in \{0.05, 0.10, 0.25, 0.50, 1.00\}$. Thus, τ determines target ambition and β the penalty per shortfall unit.

Target pressure and shortfall controllability Table 3.3 characterizes the scale of the target requirements. Increasing τ raises the total allocation target from 49.7% to 137.1% of that demand. For $\tau = 0.25$ and 0.50, average demand exceeds the total allocation target. This is necessary, but not sufficient, for meeting all targets because demand must also arrive for the appropriate items and remain allocable to the corresponding vendors. The ceiling operation in Equation (3.8) reinforces this effect: an item–vendor pair with one inventory unit has a one-unit target for every positive τ. For $\tau = 0.75$ and 0.90, the total allocation target reaches 120.7% and 137.1% of average demand, respectively, so simultaneous attainment of all targets requires above-average demand with a suitable item composition.

Figure 3.1 compares the remaining shortfall of M-MI with the lowest remaining shortfall attained by any vendor-aware policy. For each configuration, define

$$\mathrm{SR}^{\%}_{\max} = \max_{\pi \in \Pi^{\mathrm{VA}}} \mathrm{SR}^{\%}(\pi),$$

where Π^{VA} contains the vendor-aware policies (note that the maximizing policy may differ across configurations).

Increasing τ raises the amount of remaining shortfall, but sharply reduces the share that

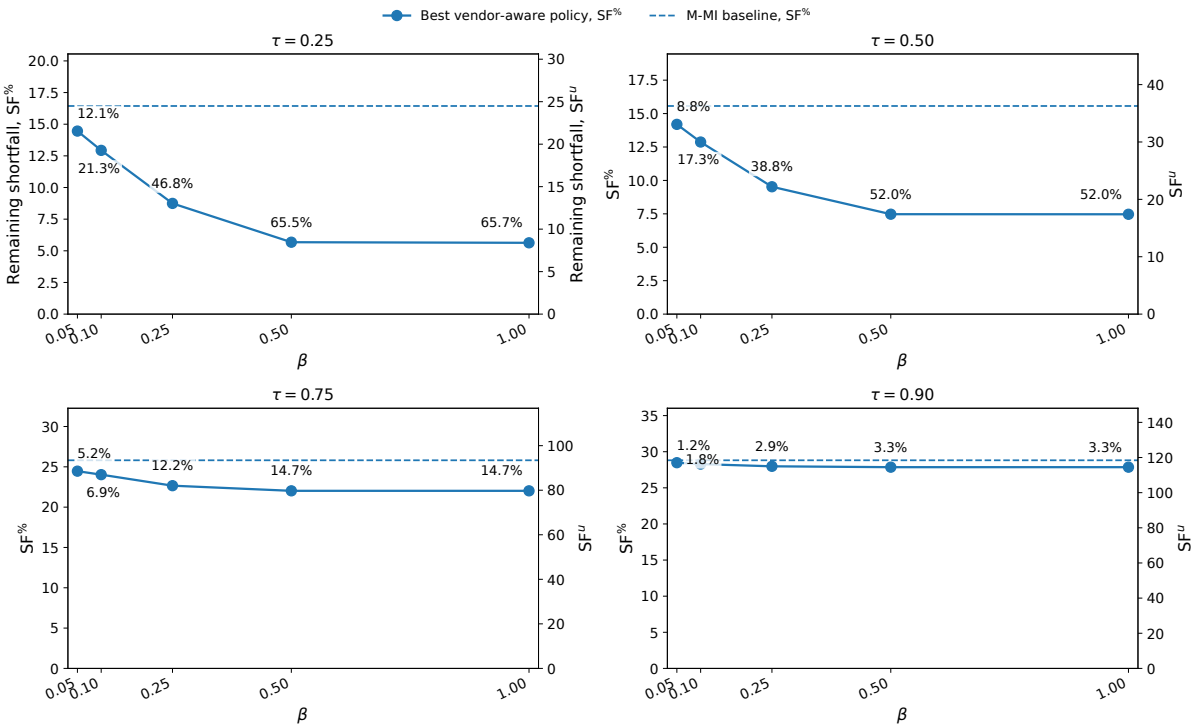


Figure 3.1: Maximum reduction in remaining target shortfall across the vendor-target stress test. The solid line reports the lowest mean remaining shortfall among vendor-aware policies and the dashed line the M-MI baseline. Labels report $\mathrm{SR}^{\%}_{\max}$.

policies can eliminate. At $\tau = 0.25$, the best vendor-aware policy removes up to 65.7% of the M-MI shortfall; at $\tau = 0.50$, the maximum is 52.0%. The corresponding values fall to 14.7% and 3.3% for $\tau = 0.75$ and 0.90. Hence, higher target ambition creates more shortfall but a progressively smaller share that is controllable through allocation policy. Increasing β initially lowers shortfall, but the gains become negligible between $\beta = 0.50$ and 1.00, indicating that the target signal has entered a saturation regime. We therefore retain $\beta = 1.00$ only for this diagnosis and exclude it from the remaining trade-off figures.

Myopic target awareness We first isolate the effect of adding target information to an otherwise myopic rule. Figure 3.2 plots $\Delta\mathrm{FC}^{\%}$(M-S) against $\mathrm{SR}^{\%}$(M-S), with point labels reporting SR^{u}(M-S). For a fixed τ, increasing β generally moves the outcome upward and to the right: M-S removes more of the M-MI shortfall, but does so by accepting a larger fulfillment-cost premium. This represents the expected increase in fulfillment cost induced by myopic vendor steering.

At $\beta = 0.05$, however, all four target levels produce a small improvement in both metrics: $\mathrm{SR}^{\%} > 0$ and $\Delta\mathrm{FC}^{\%} < 0$. This pattern is consistent with a weak target signal acting as a soft inventory-balancing mechanism. By redirecting marginal allocations toward vendors with unmet targets, M-S may preserve inventory at vendors currently favored by shipping cost for later orders in which that cost advantage is larger. The results hence show that

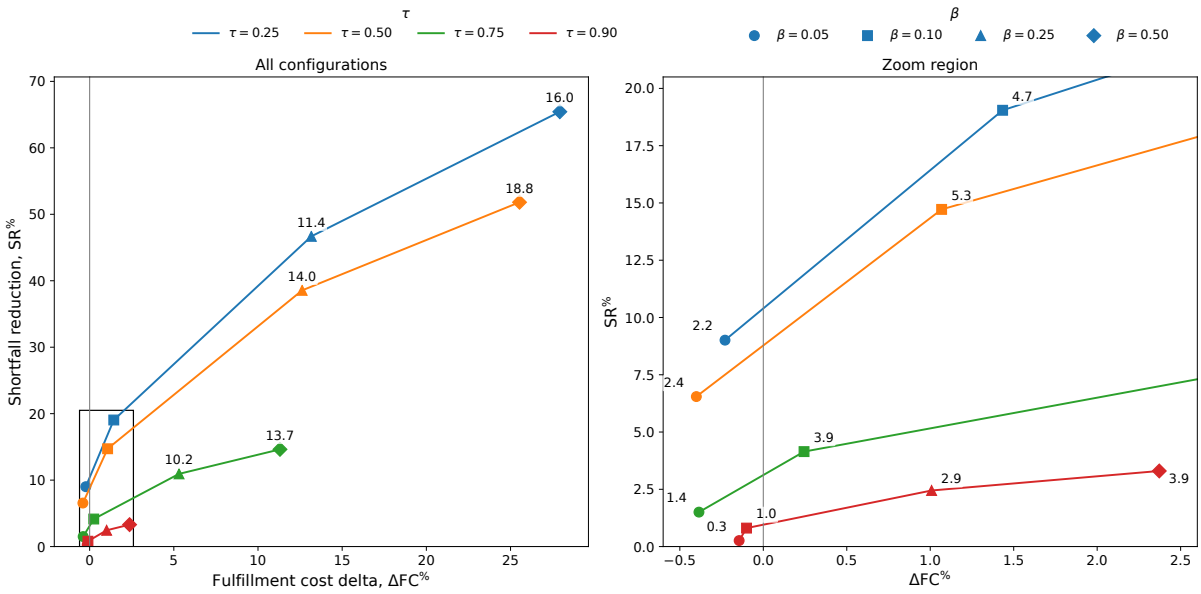


Figure 3.2: Fulfillment-cost and target-shortfall trade-off induced by myopic target awareness. Each point represents one (τ,β) configuration and reports M-S relative to M-MI. Point labels report SR^u; the right panel enlarges the low-intensity region.

limited target steering need not impose a fulfillment-cost premium. Once β reaches 0.10 or higher, the target signal generally induces sufficiently large deviations from immediate-cost allocation that the shortfall is reduced via higher fulfillment costs.

Policy trade-offs Figure 3.3 extends the comparison to all policies for $\tau \in \{0.25, 0.50\}$, which are the two τ levels that combine material baseline shortfall with substantial scope for policy-driven reduction. Each point represents one policy–β combination, and the dashed line gives the mean Pareto frontier obtained by minimizing $\Delta FC^{\%}$ and maximizing $SR^{\%}$.

The policies occupy distinct parts of the trade-off space. M-S is absent from the displayed frontiers, showing that immediate target awareness alone does not manage the trade-off efficiently. Its allocation rule values the current reduction of an unmet target, but does not distinguish between inventory that can be consumed with no future consequence and inventory that may be valuable for later demand. The two cost-function policies add opportunity-cost corrections to a similar immediate signal. Their proximity to M-S suggests that these corrections improve marginal inventory use without fundamentally changing the target response. CF-PD is generally the stronger of the two and frequently occupies the high-shortfall-reduction end of the frontier: the use of adaptive item–vendor multipliers that increase as inventory is consumed produces good results. CF-LP, by contrast, which repeatedly solves an item-specific DLP whose multi-item effects are approximated, is often dominated by other policies. Its limited additional improvement, together with the need for live optimization during execution, makes it less attractive for online use.

R-D follows a different logic. Its DLP is solved separately for each (τ,β) configuration,

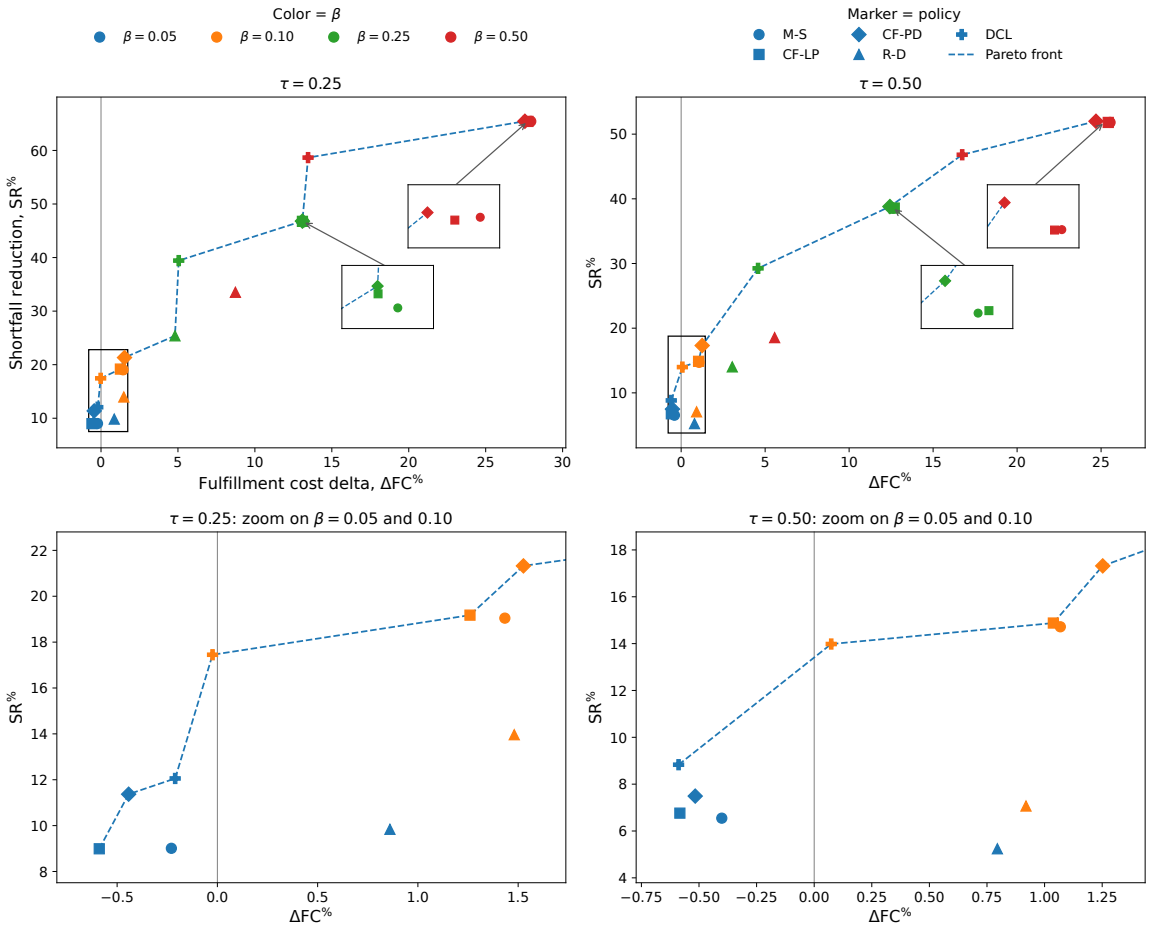


Figure 3.3: Policy trade-offs relative to M-MI for $\tau = 0.25$ and $\tau = 0.50$. Each point represents one policy–β combination. Dashed lines show the mean Pareto frontier; the lower panels and insets enlarge the four β regions.

but the resulting marginal probabilities are fixed before execution and are not continuously revised as targets are met or inventory is depleted, except through the myopic fallback and repair rules. Its correlated rounding is designed to coordinate units within multi-item orders. In the reference setting, however, inventory is already cost-aligned, item–vendor overlap is complete, and orders are small on average. These features limit the additional value of correlated assignment, while randomized deviations from the cheapest vendor can still increase fulfillment cost; accordingly, R-D generally lies away from the frontier.

DCL, in contrast, lies on the Pareto frontier for every displayed (τ,β) configuration. Its rollout labels account for downstream fulfillment costs, inventory depletion, later target-reduction opportunities, and the terminal shortfall penalty, while its state representation includes the full inventory state, remaining target requirements, delivery region, and remaining order composition. Although initialized from M-S, it can therefore learn when immediate target relief justifies its fulfillment-cost premium and when that relief should be deferred. At $(\tau,\beta) = (0.50, 0.25)$, for example, DCL removes 29.3% of M-MI's short-

fall (10.6 units) at a 4.6% fulfillment-cost increase. CF-PD removes a larger 38.8% (14.1 units), but requires a 12.4% cost increase. Thus, DCL is more cost-efficient, whereas CF-PD pursues a larger target reduction. More generally, the value of non-myopic optimization lies not simply in stronger target steering, but in making that steering more selective.

We focus on $(\tau,\beta) = (0.50, 0.25)$ in the subsequent structural analysis because it combines a significant baseline shortfall, a non-saturated penalty signal, and clear differentiation between policies.

3.5.6 Structural scenario analysis

The second experiment varies the six operational dimensions in Table 3.2, which are detailed below:

- **Vendor-wide targets**: targets are evaluated at the aggregate vendor granularity, following Equation 3.4. This is a relaxed vendor target regime in which excess allocations of one item can compensate for lower allocations of another.

- **Symmetric inventory**: replaces the cost-aligned inventory option with the symmetric option. Each item's expected demand is split equally across vendors, regardless of shipping-cost advantage. This breaks the cost alignment of the reference inventory and tests whether policies remain effective when inventory is less naturally positioned along cheap fulfillment routes.

- **Sparse item–vendor availability**: replaces the cost-aligned inventory option with the sparse-availability option. In this variant, not every vendor carries every item; each item is available only at a subset of vendors such that each vendor carries each item independently with probability 0.5. Demand for each item and region is assigned to the cheapest vendor that carries it, and inventory is generated accordingly. This creates a more structurally constrained marketplace, with fewer item–vendor substitution opportunities.

- **Uniform demand**: item demand becomes uniform. This results in all items having equal expected demand probabilities, rather than the skewed ABC demand structure (inventory is still generated from the cost-aligned procedure, but using the new uniform item-demand probabilities).

- **Larger network**: the number of items and vendors is doubled: the network changes from 20 items and 5 vendors to 40 items and 10 vendors. Average expected demand per item is kept fixed, so total expected demand doubles as a consequence of doubling the number of items. This variant tests scalability with respect to both the state and action spaces.

- **More demand**: expected demand per item is doubled, with the network size kept fixed at 20 items and 5 vendors. Since the number of items is unchanged, doubling expected demand per item also doubles total expected demand. This variant creates longer episodes and increases inventory pressure without changing the network size.

Baseline difficulty Table 3.3 shows that TDR explains much of the variation in M-MI shortfall: the larger network has TDR > 100% and the highest shortfall, whereas sparse availability and more demand have lower ratios and lower proportional shortfall. The vendor-wide variant is the main exception. It has the same total allocation target and TDR as the reference scenario, but only 4.1 rather than 36.3 shortfall units. Aggregation makes allocations across a vendor's items substitutable, so attainability depends on both the total allocation target and the flexibility with which allocations contribute to it.

Policy robustness Figure 3.4 compares the reference scenario with the six variants. Because reference inventory is cost-aligned, it favors M-MI; variants that break this alignment (symmetric inventory) or restrict substitution (sparse item–vendor availability) create more scope for forward-looking policies to improve both metrics.

Target aggregation produces the largest upward shift. Under vendor-wide targets, CF-PD removes 99.9% of baseline shortfall and DCL 86.4%, although the baseline contains only 4.1 units; pooling reduces dependence on the realized item mix. More demand also increases reducibility because the lower TDR and longer horizon provide more allocation opportunities. The larger network moves downward and rightward: TDR > 100% is combined with a larger action space and more fragmented requirements. Its best policies remove about one quarter of baseline shortfall at higher cost, still approximately 50 units[§].

Inventory and availability alter the value of coordination. Symmetric inventory breaks the alignment between stock and cheap routes, allowing DCL to preserve inventory for regions where it is most valuable; it reduces shortfall by 31.0% while lowering fulfillment cost by 1.8%. Sparse availability restricts substitution, so an early allocation can remove a later feasible option and within-order consolidation becomes more consequential. This favors order-aware policies: DCL achieves $SR^{\%} = 38.9\%$ with $\Delta FC^{\%} = -2.4\%$, while R-D's correlated rounding achieves 18.4% with −3.3%.

Uniform demand spreads opportunities across items, increasing controllable shortfall but at a higher cost premium; its balanced order types may also favor R-D's offline plan. Overall, the structural variants confirm that no policy is uniformly dominant: DCL generally provides the most cost-efficient shortfall reductions, CF-PD tends to pursue larger reductions at higher fulfillment cost, and R-D becomes more competitive when within-order coordination or offline planning is especially valuable.

[§]Note that DCL uses only one policy-iteration generation in this variant due to the increased sampling computational cost under this scenario, see Appendix 3.F

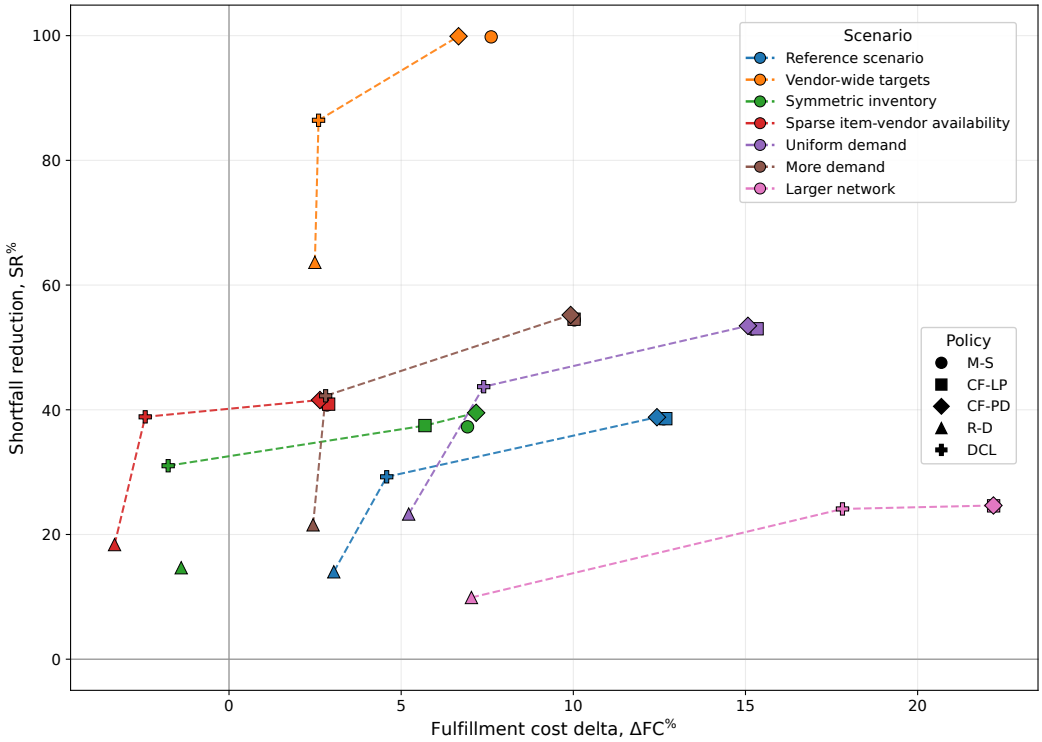


Figure 3.4: Structural-scenario policy trade-offs at $\tau = 0.50$ and $\beta = 0.25$; dashed lines show mean Pareto frontiers.

3.6. Conclusion

Online marketplaces depend on independent vendors for the assortment and inventory offered to customers. When the platform controls which vendor fulfills overlapping offers, order-allocation decisions also determine the realized business received by each vendor. Policies focused exclusively on operational efficiency may therefore weaken vendors' incentives to maintain broad assortments or commit inventory, particularly when allocations are persistently concentrated elsewhere. This paper incorporates this supply-side concern through vendor allocation targets and penalties for unmet targets.

This paper introduces the MIOFP-OM, in which a marketplace dynamically assigns arriving demand to vendor-owned inventory while balancing fulfillment costs, unfulfillment, and vendor allocation targets. We represent unmet vendor targets through end-of-season shortfall penalties and formulate the problem as an MDP-EI. The resulting formulation captures the intertemporal consequences of allocating finite inventory while incorporating vendor-side objectives that are typically absent from online fulfillment models.

The proposed formulation also applies beyond the specific marketplace setting studied here. It covers marketplaces that designate a preferred vendor through mechanisms such as a *Buy Box*, provided that customers accept the proposed vendor; settings in which only a subset of products or vendors is subject to allocation targets; and non-marketplace fulfillment systems in which a retailer wishes to enforce minimum allocation quotas across

warehouses, stores, or other fulfillment nodes. More generally, the model provides a way to study fulfillment decisions when operational efficiency must be balanced against relationship-management or allocation-commitment objectives.

We evaluate the myopic policies M-MI and M-S, with the first ignoring vendor penalties and the latter considering the immediate penalty relief, the cost-function policies CF-LP and CF-PD, the correlated-rounding policy R-D, and the RL-based policy DCL in instances motivated by a luxury fashion marketplace. The target stress test separates target ambition, controlled by τ, from penalty intensity, controlled by β. The results show that higher target ambition increases remaining shortfall but can simultaneously reduce the proportion of that shortfall that policies are able to eliminate. Thus, a large shortfall does not necessarily represent a large opportunity for policy improvement. Increasing β initially strengthens target steering, but its effect eventually saturates.

The comparison between M-MI and M-S establishes the fundamental fulfillment-cost/vendor-satisfaction trade-off. Target-aware myopic steering generally reduces shortfall by accepting higher fulfillment costs, although weak target signals can occasionally improve both outcomes. This pattern is consistent with mild target awareness acting as a soft inventory-balancing signal. Nevertheless, M-S is not Pareto-efficient in the analyzed configurations, indicating that immediate target relief alone does not adequately account for the future value of inventory.

The forward-looking policies manage this trade-off differently. DCL generally occupies the more fulfillment-cost-efficient part of the Pareto frontier, preserving substantial shortfall reductions while avoiding some of the costly allocations induced by stronger immediate target steering. CF-PD generally achieves larger shortfall reductions, but at higher fulfillment-cost premiums. R-D is less competitive in the cost-aligned reference scenario, but becomes more relevant when order-level coordination and restricted substitution are more important. These results suggest that the main benefit of non-myopic optimization is not simply stronger target steering, but more selective steering: pursuing target-reducing allocations when their future benefit justifies their immediate cost.

The scenario analysis further shows that policy performance depends on the operational structure of the marketplace. Vendor-wide targets are substantially easier to attain because allocations across items contribute to a common vendor requirement. This flexibility makes them appropriate when only aggregate vendor performance matters, whereas item–vendor targets are more suitable when the marketplace seeks to sustain assortment breadth and inventory commitment across the vendor's portfolio. Larger networks create more fragmented targets and a more complex allocation problem, reducing the proportion of shortfall that can be eliminated. Symmetric inventory and sparse item–vendor availability create additional scope for forward-looking inventory management and within-order coordination, while changes in demand intensity and item-demand concentration alter both target attainability and the relative value of adaptive and offline planning approaches.

Vendor-target steering is especially relevant for newer or maturing marketplaces seeking to strengthen their supply base and encourage vendors to expand their assortment and committed inventory. More generally, non-myopic policies are most valuable when target shortfall is both material and controllable and when current allocations meaningfully affect future fulfillment opportunities. Their practical adoption must nevertheless account

for differences in forecasting, calibration, training, and online execution requirements. The model also relies on simplifying assumptions, including committed inventory without replenishment, one fulfillment node per vendor, item-independent package costs, and common shortfall penalties. Future research could consider dynamic inventory commitment and replenishment, heterogeneous vendor or item priorities, richer target structures, and methods that improve the scalability and transferability of learned policies across larger marketplace networks.

Appendix 3.A Procedural view of the model

Algorithm 1 Selling-season evolution under policy π

Input: Initial inventory $\boldsymbol{b}^1$; policy π; exogenous input distribution Ξ; total-demand distribution Poisson(μ^D)
Sample $D \sim$ Poisson(μ^D)
$t = 1$, $S^1 = 0$
while $S^t < D$ **do**
    Sample order input $\xi^t = (j^t, L^t, \boldsymbol{d}^t) \sim \Xi$
    $\boldsymbol{x}^t \leftarrow$ Algorithm 2($\boldsymbol{b}^t, S^t, j^t, \boldsymbol{d}^t, \pi$)
    Compute order cost C^t
    Update $b_{ik}^{t+1} = b_{ik}^t - x_{ik}^t$ for all i and $k \in \mathcal{K}$
    $S^{t+1} = S^t + L^t$
    $t = t + 1$
end while
Compute terminal vendor-target cost C^{T+1}
Output: $\sum_{t=1}^{T} C^t + C^{T+1}$

Algorithm 2 Unit-level allocation of an order

Input: Inventory $\boldsymbol{b}^t$; cumulative demand S^t; order information $(j^t, \boldsymbol{d}^t)$; policy π
$\boldsymbol{w}^{t,1} = \boldsymbol{0}$, $u = 1$
for $i = 1, \ldots, n$ **do**
    for $r = 1, \ldots, d_i^t$ **do**
        $i^{t,u} = i$
        $\boldsymbol{s}^{t,u} = (\boldsymbol{b}^t, S^t, j^t, \boldsymbol{d}^t, i^{t,u}, \boldsymbol{w}^{t,u})$
        Select $k^{t,u} \in \mathcal{A}_{\boldsymbol{s}^{t,u}}$ according to policy π, conditional on any policy-specific randomization
        $w_{i^{t,u}k^{t,u}}^{t,u+1} = w_{i^{t,u}k^{t,u}}^{t,u} + 1$, with all other entries unchanged
        $u = u + 1$
    end for
end for
Output: $\boldsymbol{x}^t = \boldsymbol{w}^{t,u}$

Appendix 3.B Cost-function policy details

This appendix provides additional details on the two cost-function policies introduced in Section 3.4.2: the linear-programming-based cost-function policy, CF-LP, and the primal-dual cost-function policy, CF-PD. Both policies use opportunity-cost estimates to account for the future value of inventory when making online allocation decisions.

3.B.1 $\text{DLP}^{\text{CF-LP}}$

The CF-LP policy follows the deterministic-linear-programming approach of Acimovic and Graves (2015). At each unit-allocation decision, the policy solves a DLP for the item currently being allocated, $i^{t,u}$, and uses the dual values of the inventory constraints as estimates of the opportunity cost of consuming inventory. Since the DLP is solved for a fixed item and decision epoch, we suppress the dependence on $i^{t,u}$, t, and u in the notation.

Let b_k denote the remaining inventory of the current item at vendor k, after accounting for previous allocations in the current order. Let d_j^{rem} denote the expected remaining demand for the current item from delivery region j. The parameter $\chi^{\text{multi}} \in [0,1]$ represents the proportion of future demand expected to arrive as part of multi-unit orders. The parameter $\omega \in (0,1]$ represents the expected per-unit shipping-cost discount obtained when multiple units are shipped together in the same package. Finally, $\rho_k \in [0,1]$ approximates the probability that vendor k can also fulfill the remaining units of a multi-unit order, and therefore can serve that order without splitting.

The decision variables are:

- w_{kj}: expected single-unit demand from region j fulfilled by vendor k;
- l_{kj}: expected multi-unit demand from region j fulfilled by vendor k without splitting the order;
- m_{kj}: expected multi-unit demand from region j fulfilled by vendor k when the order is split;
- q_j^{single}: expected single-unit demand from region j left unfulfilled;
- q_j^{multi}: expected multi-unit demand from region j left unfulfilled;
- ζ_k: expected shortfall from the item–vendor allocation target of vendor k for the current item.

The DLP is:

$$\text{DLP}^{\text{CF-LP}} := \min_{\substack{w,l,m,\\ q^{single},q^{multi},\zeta}} \quad \sum_{k=1}^{K}\sum_{j=1}^{J} c_{kj}\left(w_{kj} + \omega l_{kj} + 2\omega m_{kj}\right) + c^{unf}\sum_{j=1}^{J}\left(q_j^{single} + q_j^{multi}\right) + p\sum_{k=1}^{K}\zeta_k \tag{3.9a}$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{j=1}^{J} \left(w_{kj} + l_{kj} + m_{kj} \right) \le b_k, && \forall k = 1, \dots, K, && \text{(3.9b)} \\
& \sum_{k=1}^{K} w_{kj} + q_j^{single} = d_j^{rem}(1 - \chi^{\text{multi}}), && \forall j = 1, \dots, J, && \text{(3.9c)} \\
& \sum_{k=1}^{K} \left(l_{kj} + m_{kj} \right) + q_j^{multi} = d_j^{rem} \chi^{\text{multi}}, && \forall j = 1, \dots, J, && \text{(3.9d)} \\
& l_{kj} \le \rho_k d_j^{rem} \chi^{\text{multi}}, && \forall k = 1, \dots, K, && \text{(3.9e)} \\
& && j = 1, \dots, J, \\
& \zeta_k \ge \theta_{i^{t,u}k} - \sum_{j=1}^{J} \left(w_{kj} + l_{kj} + m_{kj} \right), && \forall k = 1, \dots, K, && \text{(3.9f)} \\
& w_{kj}, l_{kj}, m_{kj} \ge 0, && \forall k = 1, \dots, K, && \text{(3.9g)} \\
& && j = 1, \dots, J, \\
& q_j^{single}, q_j^{multi} \ge 0, && \forall j = 1, \dots, J, && \text{(3.9h)} \\
& \zeta_k \ge 0, && \forall k = 1, \dots, K. && \text{(3.9i)}
\end{aligned}
$$

The objective function (3.9a) minimizes the sum of expected fulfillment costs, expected unfulfillment costs, and expected vendor-target penalties. The first term distinguishes between single-unit demand, multi-unit demand that can be fulfilled without splitting, and multi-unit demand that may require splitting. For unsplit multi-unit demand, the cost contribution is discounted by ω, because the package cost is shared across multiple units. For split multi-unit demand, the model approximates the order as requiring two packages, which gives the factor 2ω.

Constraints (3.9b) limit expected future allocations by the remaining inventory of the current item. Constraints (3.9c) assign expected single-unit demand either to vendors or to unfulfillment. Constraints (3.9d) do the same for expected multi-unit demand. Constraints (3.9e) limit the amount of multi-unit demand that can be treated as unsplit at each vendor, using ρ_k to approximate the probability that vendor k can fulfill the other units in the same order. Constraints (3.9f) define the item–vendor target shortfall for the current item, and constraints (3.9g)–(3.9i) impose nonnegativity.

Let $\nu_{i^{t,u}k}^{t,u}$ denote the dual value of constraint (3.9b) for vendor k in the $\text{DLP}^{\text{CF-LP}}$ solved at state $\boldsymbol{s}^{t,u}$. The policy uses these dual values as opportunity-cost estimates in the action rule:

$$
k_{t,u}^{\text{CF-LP}} = \underset{k \in \mathcal{A}_{\boldsymbol{s}^{t,u}}}{\arg\min} \left(g_k^{t,u} - p\delta_k^{t,u} - \nu_{i^{t,u}k}^{t,u} \right).
$$

If $k = 0$, corresponding to unfulfillment, no inventory dual is subtracted.

Parameterization in the computational experiments The expected remaining demand d_j^{rem} is computed from the item-demand and regional-demand distributions, the sampled

demand remaining in the horizon, and the current item $i^{t,u}$. The multi-unit demand share χ^{multi} and the multi-unit package-cost discount ω are computed from the order-size distribution. The vendor-level unsplit-order probabilities ρ_k estimates are computed using 1,000 simulation replications using the M-S policy, a distinct random seed from those used for evaluation, and no maximum-period cap. Note that the policy is evaluated only for item–vendor target instances, because the DLP is constructed separately for the item currently being allocated.

3.B.2 CF-PD update rule

The CF-PD policy follows the primal-dual logic of Andrews et al. (2019). Like CF-LP, it uses opportunity costs to account for the future value of inventory. However, instead of solving a forecast-based DLP, it maintains and updates multipliers online as allocations are made.

Let $\lambda_{ik}^{t,u}$ denote the primal-dual multiplier associated with the inventory of item i at vendor k before allocating unit u of order t. These multipliers are initialized at zero:

$$\lambda_{ik}^{1,1} = 0, \qquad \forall i = 1,\dots,n,\ k \in \mathcal{K}.$$

For the current unit $i^{t,u}$, define $\mathcal{F}^{t,u}$ as the set of feasible real vendors:

$$\mathcal{F}^{t,u} = \left\{k \in \mathcal{K} : b^{t}_{i^{t,u}k} - w^{t,u}_{i^{t,u}k} > 0\right\}.$$

If $\mathcal{F}^{t,u} = \varnothing$, the unit is assigned to the unfulfillment action $k = 0$. Otherwise, the policy chooses among vendors in $\mathcal{F}^{t,u}$.

Because the primal-dual rule is naturally formulated in terms of rewards, while the MIOFP-OM is formulated as a cost-minimization problem, the policy first converts costs into reward-like quantities. For each $k \in \mathcal{F}^{t,u}$, define the shifted cost

$$\tilde{g}_k^{t,u} = g_k^{t,u} - p\delta_k^{t,u} + p,$$

where $g_k^{t,u}$ is the immediate fulfillment cost defined in the myopic-policy subsection, and $\delta_k^{t,u}$ indicates whether allocating the current unit to vendor k reduces a still-unmet vendor target. For item–vendor targets,

$$\delta_k^{t,u} = \begin{cases} 1, & \text{if } A^{t,u}_{i^{t,u}k} < \theta_{i^{t,u}k}, \\ 0, & \text{otherwise,} \end{cases} \qquad k \in \mathcal{F}^{t,u}.$$

For vendor-wide targets,

$$\delta_k^{t,u} = \begin{cases} 1, & \text{if } A^{t,u}_{k} < \theta_{k}, \\ 0, & \text{otherwise,} \end{cases} \qquad A_k^{t,u} = \sum_i A_{ik}^{t,u}.$$

The term $-p\delta_k^{t,u}$ reflects the avoided terminal penalty from reducing a target shortfall, while the final $+p$ term keeps the shifted cost nonnegative and comparable across vendors.

Let

$$\bar{g}^{t,u} = \frac{1}{|\mathcal{F}^{t,u}|} \sum_{k \in \mathcal{F}^{t,u}} \tilde{g}_k^{t,u}$$

be the average shifted cost among feasible real vendors. The reward signal is defined as

$$R_k^{t,u} = \max\left\{ \exp\left(\frac{\bar{g}^{t,u} - \tilde{g}_k^{t,u}}{\max\{\bar{g}^{t,u}, 1\}} \right), R^{min} \right\}, \qquad k \in \mathcal{F}^{t,u}.$$

Thus, vendors with lower shifted cost receive larger reward values, while $R^{min} > 0$ ensures a strictly positive lower bound.

The CF-PD action rule is

$$k_{t,u}^{\text{CF-PD}} = \arg\max_{k \in \mathcal{F}^{t,u}} \left(R_k^{t,u} - \lambda_{i^{t,u}k}^{t,u} \right).$$

After selecting vendor $k_{t,u}^{\text{CF-PD}}$, only the multiplier associated with the allocated item–vendor pair is updated. Let $k^\star = k_{t,u}^{\text{CF-PD}}$ and $i^\star = i^{t,u}$. The update is

$$\lambda_{i^\star k^\star}^{t,u+1} = \lambda_{i^\star k^\star}^{t,u} \left(1 + \frac{1}{\alpha_1 \bar{b} + \alpha_2} \right) + \beta^{pd} \frac{R_{k^\star}^{t,u}}{\alpha_1 \bar{b} + \alpha_2},$$

while all other multipliers remain unchanged:

$$\lambda_{ik}^{t,u+1} = \lambda_{ik}^{t,u}, \qquad (i,k) \neq (i^\star, k^\star).$$

Here α_1, α_2, and β^{pd} are update parameters, and $\bar{b}$ is a scale parameter based on the initial inventory. The update increases the multiplier of the item–vendor pair that has just been consumed, thereby making future allocations to the same pair less attractive.

Parameter calibration in the computational experiments. In the computational experiments, the parameters of CF-PD are calibrated separately for each generated instance. We use a simulation-based grid search over

$$\alpha_1 \in \{0.25, 0.5, 1.0, 1.5, 2.0\}, \qquad \alpha_2 \in \{0.0, 1.0, 2.0, 5.0\}, \qquad \beta^{pd} \in \{0.01, 0.05, 0.10, 0.25\}.$$

For each candidate parameter vector, we simulate 100 calibration trajectories with a seed distinct from those used for evaluation and without a maximum-period cap. The selected parameter vector is the one with the lowest average total cost over the calibration trajectories. Ties are broken toward the more conservative update, corresponding to the smaller value of β^{pd}.

We set $R^{min} = \exp(-1)$ and define $\bar{b}$ as the minimum positive initial inventory across item–vendor pairs. Unlike CF-LP, the CF-PD policy does not require explicit forecasts of future demand and is evaluated for both item–vendor and vendor-wide target formulations.

Appendix 3.C Dilate rounding policy details

This appendix provides additional details on the rounding policy R-D introduced in Section 3.4.3. The policy first solves an offline DLP, DLP^R, which produces target fulfillment probabilities. These probabilities are then used by a correlated rounding procedure to make online allocation decisions.

3.C.1 DLP^R

Let $\mathcal{Q}^R$ denote the set of order types represented in the rounding-policy DLP. An order type $q \in \mathcal{Q}^R$ specifies the items contained in an order. Let h_j^q denote the probability that an order of type q arrives from delivery region j, with $\sum_{q,j} h_j^q \leq 1$. Let T^R denote the expected number of order arrivals considered by the offline DLP. Then $T^R h_j^q$ is the expected number of orders of type q from region j over the horizon represented in the DLP.

The decision variables are:

- r_{kij}^q: proportion of demand for item $i \in q$, in orders of type q from region j, assigned to vendor k;
- v_{kj}^q: proportion of orders of type q from region j for which vendor k is used to fulfill at least one item;
- ζ_{ik}: shortfall from the item–vendor target of item i at vendor k, for item–vendor target instances;
- ζ_k: shortfall from the aggregate target of vendor k, for vendor-wide target instances.

Vendor $k = 0$ represents unfulfillment and has infinite inventory. Real vendors are indexed by $k = 1, \ldots, K$. The offline DLP for item–vendor targets is:

$$\begin{aligned}
\text{DLP}^R := \min_{r,v,\zeta} \quad & T^R \sum_{q \in \mathcal{Q}^R} \sum_{j=1}^{J} \left[\sum_{k=1}^{K} c_{kj} v_{kj}^q + c^{unf} \sum_{i \in q} r_{0ij}^q \right] \\
& + p \sum_{i=1}^{n} \sum_{k=1}^{K} \zeta_{ik} && && \text{(3.10a)} \\
\text{s.t.} \quad & T^R \sum_{j=1}^{J} \sum_{\substack{q \in \mathcal{Q}^R: \\ i \in q}} h_j^q r_{kij}^q \leq b_{ik}, && \forall i = 1, \ldots, n, \; k \in \mathcal{K}, && \text{(3.10b)} \\
& \sum_{k=0}^{K} r_{kij}^q = 1, && \forall q \in \mathcal{Q}^R, \; j = 1, \ldots, J, \; i \in q, && \text{(3.10c)} \\
& v_{kj}^q \geq r_{kij}^q, && \forall q \in \mathcal{Q}^R, \; j = 1, \ldots, J, \; i \in q, \; k = 1, \ldots, K, && \text{(3.10d)}
\end{aligned}$$

$$\zeta_{ik} \geq \theta_{ik} - T^R \sum_{j=1}^{J} \sum_{\substack{q \in Q^R: \\ i \in q}} h_j^q r_{kij}^q, \qquad \forall i = 1, \ldots, n, \; k \in \mathcal{K}, \tag{3.10e}$$

$$r_{kij}^q \geq 0, \qquad \forall q \in Q^R, \; j = 1, \ldots, J, \; i \in q, \; k = 0, \ldots, K, \tag{3.10f}$$

$$v_{kj}^q \geq 0, \qquad \forall q \in Q^R, \; j = 1, \ldots, J, \; k = 1, \ldots, K, \tag{3.10g}$$

$$\zeta_{ik} \geq 0, \qquad \forall i = 1, \ldots, n, \; k \in \mathcal{K}. \tag{3.10h}$$

The objective function (3.10a) minimizes expected fulfillment cost, expected unfulfillment cost, and the vendor-target penalty. The fulfillment-cost term uses v_{kj}^q because the cost c_{kj} is incurred once when vendor k is used to fulfill at least one item in an order of type q from region j. The unfulfillment-cost term is incurred per unfulfilled unit, represented by assignments to $k = 0$.

Constraints (3.10b) ensure that expected allocations of each item to each real vendor do not exceed the inventory initially committed by that vendor. Constraints (3.10c) ensure that every item in every represented order type is assigned either to a real vendor or to the unfulfillment action. Constraints (3.10d) link item-level assignment probabilities to vendor-use probabilities. Constraints (3.10e) define the item–vendor target shortfalls, and constraints (3.10f)–(3.10h) impose nonnegativity.

For vendor-wide targets, the DLP is identical except that constraints (3.10e) are replaced by

$$\zeta_k \geq \theta_k - T^R \sum_{i=1}^{n} \sum_{j=1}^{J} \sum_{\substack{q \in Q^R: \\ i \in q}} h_j^q r_{kij}^q, \qquad \forall k = 1, \ldots, K, \tag{3.11}$$

and the vendor-target term in the objective becomes

$$p \sum_{k=1}^{K} \zeta_k.$$

Thus, for vendor-wide targets, target shortfall is measured at the aggregate vendor level.

Let $r_{kij}^{q,*}$ denote an optimal solution of DLP^R. During online execution, these values are used as marginal fulfillment probabilities by the rounding policies.

3.C.2 Dilate rounding

Consider an arriving order t of type $q \in Q^R$ from delivery region j^t. For each item $i \in q$, define

$$r_{ki} = r^{q,*}_{kij^t}, \qquad k = 0, \ldots, K.$$

The correlated rounding scheme uses these probabilities to assign each item in the order to a vendor. The unfulfillment action $k = 0$ is included in the probability vector but does not correspond to a physical shipment. For each k with $\max_{i \in q} r_{ki} > 0$, define

$$\bar{r}_k = \max_{i \in q} r_{ki}.$$

The R-D policy draws an independent exponential opening time E_k for each k, with mean $1/\bar{r}_k$. Item i observes vendor k opening at the dilated time

$$X_{ki} = \frac{\bar{r}_k}{r_{ki}} E_k,$$

with $X_{ki} = \infty$ if $r_{ki} = 0$. The rounded assignment is

$$\hat{k}_i = \arg\min_{k \in \{0,\ldots,K\}} X_{ki}.$$

This construction preserves the marginal probabilities:

$$\Pr(\hat{k}_i = k) = r_{ki}.$$

Because the same opening times E_k are shared across items in the order, the assignments are positively correlated, making it more likely that multiple items are assigned to the same vendor.

Inventory feasibility and repair The DLP probabilities are computed offline and therefore do not necessarily account for the realized inventory state at the moment an order arrives. If the rounded assignment $\hat{k}_i$ is a real vendor with remaining inventory for item i, the unit is assigned to $\hat{k}_i$. If $\hat{k}_i = 0$, the unit is assigned to the unfulfillment action. If $\hat{k}_i > 0$ but vendor $\hat{k}_i$ no longer has inventory of item i, the rounded assignment is infeasible. In these cases, we apply the policy M-S to the current unit to repair the assignment. If an arriving order type q is not represented in Q^R, the rounding policy does not have DLP probabilities for that order. In that case, the policy follows M-S.

3.C.3 Parameterization in the computational experiments

In the computational experiments, Q^R includes only order types that satisfy two filters. First, the order type must have a size no larger than the maximum order-size cutoff, which is set to 6; all order types up to that cap are eligible for inclusion. Second, order types with probability lower than 10^{-5} are excluded from Q^R. This probability threshold reduces the

size of DLP^R by removing extremely rare order types that would have a negligible impact on expected costs but could substantially increase solution time. The order-type probabilities h_j^q are computed from the regional-demand, order-size, and item-demand distributions used in the instance generation, after applying these filters.

The offline DLP is solved once for each scenario and target formulation. The resulting probabilities $r_{kij}^{q,*}$ are then fixed and reused throughout the online simulations.

Appendix 3.D DCL Implementation Details

This appendix provides additional details on the implementation of DCL in the computational experiments. The algorithm follows the approximate-policy-iteration framework of Temizöz et al. (2025). At each generation, DCL samples states, identifies simulation-based improved actions through rollout simulations, and trains a NN classifier to approximate the resulting state–action mapping.

3.D.1 Algorithmic overview

Algorithm 3 presents the DCL procedure used in this work. The algorithm takes as input the MDP-EI model $\mathcal{M}$, an initial policy π_0, a NN structure N_ψ, and a set of sampling and training parameters. At each approximate policy-iteration step g, the algorithm constructs a dataset

$$\mathcal{D}_g = \left\{(\boldsymbol{s}_m, \hat{\pi}_g^+(\boldsymbol{s}_m)) : m = 1, \ldots, N\right\},$$

where $\hat{\pi}_g^+(\boldsymbol{s}_m)$ is the simulation-based improved action for sampled state $\boldsymbol{s}_m$. A NN classifier is then trained on this dataset to obtain the next policy π_{g+1}.

Algorithm 3 DCL

Input: MDP-EI model $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{W}, \Xi, f, C, \alpha, \boldsymbol{s}_0 \rangle$; initial policy π_0; NN structure N_ψ; number of approximate policy-iteration steps G; number of sampled states N; number of exogenous scenarios per state–action pair M; rollout horizon H; number of workers N^{wrk}
for $g = 0, 1, \ldots, G-1$ **do**
  $\mathcal{D}_g \leftarrow \emptyset$
  for each worker $= 1, \ldots, N^{\text{wrk}}$ **in parallel do**
    $\boldsymbol{s}_1 \leftarrow \boldsymbol{s}_0$
    Generate an exogenous scenario $\boldsymbol{\xi}$ with length $\lceil N/N^{\text{wrk}} \rceil$
    for $m = 1, \ldots, \lceil N/N^{\text{wrk}} \rceil$ **do**
      $\hat{\pi}_g^+(\boldsymbol{s}_m) \leftarrow \textsc{Simulator}(\mathcal{M}, \boldsymbol{s}_m, \pi_g, M, H)$
      Add $(\boldsymbol{s}_m, \hat{\pi}_g^+(\boldsymbol{s}_m))$ to $\mathcal{D}_g$
      $\boldsymbol{s}_{m+1} \leftarrow f(\boldsymbol{s}_m, \hat{\pi}_g^+(\boldsymbol{s}_m), \xi_m)$
    end for
  end for
  $\pi_{g+1} \leftarrow \textsc{Classifier}(N_\psi, \mathcal{D}_g)$
end for
Output: $\pi_1, \ldots, \pi_G$

In this finite-horizon setting, each worker initializes its sampled trajectory at the initial state $\boldsymbol{s}_0$. The worker then generates an independent exogenous scenario and advances through the state space by applying the simulation-based improved action at each sampled state. This produces sampled states along finite selling-season trajectories without requiring any additional start-state sampling parameter.

The procedure Simulator estimates the improved action in a sampled state. For each feasible action, it simulates rollouts of length H: the first action is fixed to the action under evaluation, and subsequent decisions follow the current policy π_g. To compare actions more accurately under stochastic demand, DCL uses common random numbers, evaluating competing actions under the same exogenous scenarios. It also uses sequential halving to allocate more simulation effort to actions that appear more promising. The final surviving action is used as the simulation-based label $\hat{\pi}_g^+(\boldsymbol{s})$.

The procedure Classifier trains a NN classifier on $\mathcal{D}_g$. The NN outputs one score for each action $k \in \mathcal{K}_0$, where $k = 0$ denotes unfulfillment and $k \in \mathcal{K}$ denotes a real vendor. Infeasible actions are masked when applying the policy, and the action with the highest feasible score is selected.

3.D.2 State features

The DCL policy is trained on a feature representation of the unit-allocation state $\boldsymbol{s}^{t,u}$. The feature vector is designed to capture expected remaining demand, the current order information, current inventory availability, and target pressure. The features used in the computational experiments are:

- **Expected remaining demand by item.** A vector of length n, where entry i is

  $$\max\{\mu^D - S^t, 0\}\, O_i^{item}.$$

  Here, μ^D is the expected total seasonal demand, S^t is the cumulative demand observed before the current order, and O_i^{item} is the probability that a demand unit corresponds to item i. This feature approximates the expected number of remaining demand units for each item.

- **Delivery region.** A one-hot vector of length J, with entry j^t equal to one and all other entries equal to zero.

- **Current inventory.** The current inventory level b_{ik}^t for every item–vendor pair (i,k), added vendor by vendor. This gives the NN information about the full inventory state, not only the inventory of the item currently being allocated.

- **Action-level adjusted allocation cost.** A vector over real vendors $k = 1, \ldots, K$. For each vendor, the value equals the incremental package cost of assigning the current unit to that vendor, net of any immediate target relief:

  $$\tilde{g}_k^{t,u} = c_{kj^t}\, \mathbf{1}\{k \text{ has not yet been used in order } t\} - p\, \mathbf{1}\{\Delta_k^{t,u} > 0\}.$$

  where $\Delta_k^{t,u}$ denotes the remaining allocation slack for vendor k. Thus, if assigning the current unit to vendor k reduces an unmet allocation target, the feature subtracts the vendor-target penalty coefficient p.

- **Allocation slack by vendor.** A vector over real vendors $k = 1, \ldots, K$. For item–vendor targets, the slack is

  $$\Delta_k^{t,u} = \theta_{i^{t,u}k} - A_{i^{t,u}k}^{t,u}.$$

  For vendor-wide targets, the slack is

  $$\Delta_k^{t,u} = \theta_k - A_k^{t,u}.$$

  Positive values indicate that the corresponding target has not yet been met.

- **Current-item inventory by vendor.** A vector over real vendors $k = 1, \ldots, K$, with entries $b_{i^{t,u}k}^t$, representing the amount of inventory of the current item available at each vendor.

- **Current item identity.** A one-hot vector of length n, with entry $i^{t,u}$ equal to one and all other entries equal to zero.

- **Remaining order composition.** A vector of length n, where each entry records how many units of that item remain to be allocated after the current unit within the same order.

The feature representation, therefore, combines expected remaining demand, current order information, full inventory availability, item-specific inventory availability, and target-pressure information.

3.D.3 Sampling and training parameters

Table 3.4 summarizes the DCL sampling, simulation, and neural-network parameters used in the computational experiments.

$D_{0.90}^{\ell}$ denotes an approximation of the 90th percentile of the total seasonal demand distribution for instance ℓ. Since total seasonal demand is modeled as $D^{\ell} \sim \text{Poisson}(\mu^{\ell})$, we approximate this percentile by the normal approximation

$$D_{0.90}^{\ell} = \left\lceil \mu^{\ell} + z(0.90)\sqrt{\mu^{\ell}} - 0.5 \right\rceil,$$

where $z(0.90) = 1.281551565545$ is the 90th percentile of the standard normal distribution.

The rollout horizon is set to $H = \lceil D_{0.90}^{\ell} \rceil$, so rollouts are long enough to reach the end of the selling season in a large share of simulated demand realizations. This is important because vendor-target penalties are incurred only at the terminal state. If rollouts stop before the end of the season, these terminal costs are not observed when generating simulation-based action labels.

The number of sampled states is set to

$$N = \left\lceil H \times N^{\text{wrk}} \right\rceil,$$

Table 3.4: Specifications for DCL sampling, training, and NN structure N_ψ

Sampling and Simulation		**NN Structure N_ψ**	
Property	**Value**	**Property**	**Value**
Initial policy	π_0 = M-S	Network type	MLP
Rollout horizon	$H = \left\lceil D^{\ell}_{0.90} \right\rceil$	Hidden layers	{`firstLayer`, 32}
Number of exogenous scenarios	$M = 100$	Optimizer	Adam
Number of samples	$N = \left\lceil H \times N^{\text{wrk}} \right\rceil$	Mini-batch size	64
Early stopping patience	10	Output size	$K+1$
Number of workers	N^{wrk} = `dclThreadCount`		
Sampling probability	1.0		
Sequential halving	Enabled		
Retrain last generation only	False		
Number of approximate policy-iteration steps: $G = 3$ **(except larger network variant, where** $G = 1$**)**			

where N^{wrk} is the number of DCL worker threads. The sampling probability is set to one, so each worker collects approximately one sampled state per rollout period. The number of exogenous scenarios is set to 100.

The initial policy is M-S. This provides a simple but informative starting point because it accounts for immediate fulfillment cost and immediate vendor-target relief. The learned DCL policy then improves on this initial policy through simulation-based approximate policy iteration.

The NN architecture is a fully connected multi-layer perceptron. The first hidden layer size, `firstLayer`, is set to 128 in all scenarios except the larger network, which has size 256. This is to accommodate the significantly larger input layer of that scenario (580 versus 205 inputs in all other scenarios). The second hidden layer has 32 neurons across all configurations. The output layer has $K + 1$ units, one for each real vendor and one for the unfulfillment action. Although the feature vector does not include an explicit unfulfillment-cost entry in the action-level cost vector, the unfulfillment action is represented in the output layer and remains available whenever selected by the learned policy subject to the feasible-action logic.

Training uses the Adam optimizer, mini-batches of size 64, and early stopping with patience 10. The loss function is cross-entropy between the NN output and the simulation-based action labels. At inference time, infeasible actions are masked and the feasible action with the highest output score is selected.

The number of approximate policy-iteration steps is set to $G = 3$, producing π_1, π_2, and π_3. The reported DCL results use the last available generation. For the larger-network variant, only $G = 1$ is used because of its higher computational cost. The corresponding training times are reported in Appendix 3.F.

Appendix 3.E Detailed instance generation procedure

This appendix details each component of the instance generation procedure. Table 3.5 summarizes the distinction between fixed instance-construction rules and stress-test and scenario configuration knobs, and indicates where each component is defined in the remainder of this appendix.

3.E.1 Network and fulfillment-cost construction

The fulfillment network is generated from a set of candidate vendor countries, candidate delivery-region countries, country-level population data, and a country-to-country fulfillment cost matrix. Country populations are obtained from Eurostat's population-on-1-January dataset for 2024 (Eurostat, 2024). These population values are used both to select the largest candidate countries and to construct the regional demand distribution.

Fulfillment costs use European Commission parcel-delivery price data (Commission, 2025). For each origin–destination country pair, we use the lowest reported tariff for a "1kg track-and-trace parcel delivery" service, domestic or cross-border, as the corresponding country-to-country fulfillment cost. The resulting cost c_{kj} is therefore the lowest available tariff for the specified service from vendor country k to delivery country j. The experiments use the 2025 tariff data, corresponding to prices applicable as of 1 January 2025. For domestic routes, we apply one adjustment. If the reported domestic tariff for a vendor country exceeds the minimum cross-border tariff from that vendor country to any selected delivery country, the domestic cost is set to 95% of that minimum cross-border tariff. This adjustment prevents domestic fulfillment from being dominated by the cheapest cross-border alternative when the reported domestic tariff is unusually high relative to the cross-border tariffs used in the experiment.

The number of delivery regions is fixed to $J = 30$. For a given number of vendors K, the implementation selects the K candidate vendor countries with the largest populations. The fulfillment cost matrix $\boldsymbol{c}$ is then obtained by restricting the full country-to-country cost matrix to the selected destination–origin pairs, so that c_{kj} denotes the cost of fulfilling an order from vendor k to delivery region j.

The regional demand distribution is proportional to the population of the delivery regions. Let Pop_j denote the 2024 Eurostat population of delivery region (country) j. Then

$$\mathbb{P}(j) = \frac{\mathrm{Pop}_j}{\sum_{j'=1}^{J} \mathrm{Pop}_{j'}}, \qquad j = 1, \ldots, J.$$

This distribution is used both to sample the delivery region of arriving orders and to construct the expected item–vendor demand shares used for inventory initialization.

The unfulfillment cost is scaled relative to the most expensive fulfillment route. Specifically,

$$c^{unf} = \gamma^{unf} \max_{k,j} c_{kj},$$

Table 3.5: Instance-generation components and their role in the computational experiments. The numbered columns refer, respectively, to network and fulfillment-cost construction, item-level demand and total seasonal demand, order-arrival process, inventory initialization, and allocation targets.

Role	**3.E.1**	**3.E.2**	**3.E.3**	**3.E.4**	**3.E.5**
Fixed construction rules	Country data; parcel tariffs; number of delivery regions J; cost matrix $\boldsymbol{c}$; scaling for c^{unf}.	Population-based regional weights.	Geometric order-size distribution.	Newsvendor-type adjustment, conditional on demand shares.	Item–vendor construction; vendor-wide aggregation.
Stress-test configuration	–	–	–	–	Target ambition τ; penalty intensity β.
Scenario configuration	Network size (n, K).	Item-demand option; demand intensity $\bar{\mu}$.	–	Inventory-initialization option.	Target granularity

where γ^{unf} is the demand-unfulfillment coefficient. We fix $\gamma^{unf} = 2$, so unfulfillment is substantially penalized relative to physical fulfillment.

3.E.2 Item-level demand and total seasonal demand

Item-level demand is generated according to the item-demand option specified for each experimental scenario. We consider two item-demand options: an ABC demand option and a uniform-demand option. In both cases, the expected total seasonal demand in instance ℓ is set to

$$\mu^{\ell} = \lfloor n\bar{\mu} \rceil,$$

where n is the number of items, $\bar{\mu}$ is the average expected demand per item, and $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.

Under the ABC demand option, item-demand probabilities are defined using rank-based weights. Specifically, item i receives weight

$$\kappa_i = \frac{1}{i}, \qquad i = 1, \ldots, n,$$

and the item-demand distribution is

$$\mathbb{P}^{\ell}(i) = \frac{\kappa_i}{\sum_{r=1}^{n} \kappa_r}, \qquad i = 1, \ldots, n.$$

This construction generates a skewed demand distribution, where lower-ranked items account for a larger share of expected demand.

Under the uniform-demand option, all items have the same demand probability:

$$\mathbb{P}^{\ell}(i) = \frac{1}{n}, \qquad i = 1, \ldots, n.$$

At the beginning of each simulated selling season, total seasonal demand is sampled as

$$D^{\ell} \sim \text{Poisson}(\mu^{\ell}).$$

The selling season ends once the cumulative number of demand units observed reaches or exceeds this sampled total demand.

3.E.3 Order-arrival process

Each order is generated by independently sampling its delivery region, size, and item composition. The delivery region is sampled from the population-weighted regional distribution described above.

The order size L^t is sampled from a geometric distribution with support $\{1, 2, \ldots\}$ and mean μ^{order}:

$$L^t \sim \text{Geom}\left(\frac{1}{\mu^{order}}\right),$$

where $\mu^{order} = 5/3$. Conditional on L^t, each unit in the order is independently assigned to an item according to the item-demand distribution $\mathbb{P}^\ell(i)$.

The resulting list of items is sorted before being processed. This sorting does not change the order composition, but it makes the within-order unit-allocation sequence deterministic conditional on the sampled item multiset.

3.E.4 Inventory initialization

Initial inventory is generated from expected item–vendor demand shares. Let η^ℓ_{ik} denote the expected share of total seasonal demand corresponding to item i and vendor k in instance ℓ. The implementation supports three inventory-generation options.

Under the cost-aligned option, vendor demand shares are based on cost-based regional capture. Each delivery region is assigned to the vendor with the lowest fulfillment cost for that region. The capture share of vendor k is then

$$\phi_k = \sum_{j:\, k=\arg\min_{k'} c_{k'j}} \mathbb{P}(j).$$

The item–vendor share is

$$\eta^\ell_{ik} = \mathbb{P}^\ell(i)\phi_k.$$

Thus, inventory is positioned according to the demand each vendor would be expected to serve under a purely cost-based regional assignment.

Under the symmetric option, expected item demand is split uniformly across vendors:

$$\eta^\ell_{ik} = \frac{\mathbb{P}^\ell(i)}{K}.$$

Under the sparse-availability option, the implementation generates a sparse item–vendor availability structure, using the same procedure as Jasin and Sinha (2015) and Ma (2023). For each item, each vendor independently carries that item with probability 0.5. If no vendor is sampled as carrying the item, the item is assigned to the vendor with the lowest population-weighted average fulfillment cost. For each item–region pair, the expected demand share is assigned to the cheapest vendor among those carrying that item.

Given η^ℓ_{ik}, initial inventory is set using a Newsvendor-type safety adjustment:

$$b^{1,\ell}_{ik} = \left\lceil \mu^\ell \eta^\ell_{ik} + z^{\text{stock}} \sqrt{\mu^\ell \eta^\ell_{ik}(1-\eta^\ell_{ik})} \right\rceil.$$

We set $z^{\text{stock}} = 0.5$ in all experimental configurations. This construction places more inventory at item–vendor pairs expected to serve more demand, while adding a safety adjustment that increases with demand uncertainty.

3.E.5 Allocation targets and penalty

Allocation targets are generated from the initial inventory matrix and the target-ambition parameter τ. The primitive item–vendor target is

$$\theta_{ik}^{\ell} = \left\lceil \tau b_{ik}^{1,\ell} \right\rceil, \qquad i = 1, \ldots, n,\ k \in \mathcal{K}.$$

When item–vendor targets are used, shortfalls are evaluated separately for each item–vendor pair (Equation 3.3). For vendor-wide targets, the item–vendor targets are first aggregated into a vendor-level target,

$$\theta_{k}^{\ell} = \sum_{i=1}^{n} \theta_{ik}^{\ell}, \qquad k \in \mathcal{K},$$

and shortfalls are evaluated only at the vendor level (Equation 3.4).

Similar to the unfulfillment cost c^{unf}, the vendor-target penalty coefficient is scaled relative to the largest fulfillment cost:

$$p = \beta \max_{k,j} c_{kj},$$

where β is the vendor-target penalty coefficient varied in the experiments.

3.E.6 Generated instance

For a fixed experimental configuration, the generated instance is characterized by

$$\mathcal{I}^{\ell} = \left\langle n, K, J, \boldsymbol{c}, c^{unf}, p, \mu^{\ell}, O^{\ell}, \boldsymbol{b}^{1,\ell}, \boldsymbol{\theta}^{\ell} \right\rangle.$$

The fixed construction rules in Table 3.5 determine how these objects are computed from the selected configuration. The scenario-configuration dimensions determine the network, demand, and inventory components, while the stress-test configuration determines the target structure and vendor-target penalty coefficient.

Appendix 3.F Policy preparation and online execution time

The main paper compares policies through their fulfillment-cost and target-shortfall performance. This section complements that analysis by reporting implementation-specific offline preparation and online execution times. All computational experiments were run on AMD Rome CPU nodes using 128 CPU cores and 224 GiB of RAM. The policies and simulation environment were implemented using the DynaPlex software library (Akkerman et al., 2023). Linear programs were solved using Gurobi 12.0.3. Since preparation and execution times depend on both the computing environment and the implementation, the values reported in this appendix should be interpreted as implementation-specific measurements rather than complexity estimates.

Table 3.6 reports the recorded preparation times for R-D and DCL, the two policies with separately timed offline optimization or training stages. For R-D, the reported time covers construction and solution of its offline DLP. For DCL, it corresponds to the total training time for the available generations. For most standard-size scenarios, randomized-policy preparation is below two minutes, although the uniform-demand variant is an exception. This is due to the fact that, with uniform demand, fewer order types are pruned following the probability cut-off of 10^{-5}, leading to a significantly larger LP to solve (see Appendix 3.C.3). DCL requires the largest offline preparation effort: around one to two hours in standard-size scenarios, and substantially more in the larger-network and more-demand variants, since they require simulating longer horizons and, in larger-network scenarios specifically, evaluating more allocation choices. It is important to note that the reported time for the larger-network scenario corresponds to a single iteration, whereas all other times correspond to three.

CF-PD and CF-LP preparation times are negligible. For CF-PD, calibration for each scenario is performed by evaluating 80 parameter combinations on 100 trajectories each, for a total of 8,000 calibration trajectories. For CF-LP, the preparation step consists of running 1,000 trajectories of M-S to estimate ρ. We refer to Table 3.7, which reports the per-step execution time of CF-PD and M-S.

Finally, Table 3.7 reports online execution times. We report the elapsed time per allocation step, which approximates the time required for a policy to decide how to allocate one unit of demand. The standard-size and more-demand scenarios exhibit similar per-step ranges, so they are summarized together, while the larger-network scenario is reported separately. All performance comparisons in Section 3.5 use the common set of 1,000 trajectories defined in the main paper. For runtime measurement only, all policies except CF-LP were timed over 10,000 trajectories, whereas CF-LP was timed over 1,000 trajectories because of its substantially higher online execution cost.

The results show that the myopic, randomized, and CF-PD policies all execute very quickly online, with per-step times ranging from sub-microseconds to microseconds in the recorded implementation. DCL is slower than these policies but remains relatively fast, since online execution only requires evaluating the learned policy. The larger-network variant has slightly higher per-step times across policy classes, because each allocation decision involves a larger set of vendors, and therefore more feasible allocation choices. By contrast, CF-LP is several orders of magnitude slower per step, reflecting the fact that

Table 3.6: Recorded preparation-time components by policy class. Times are reported in minutes. The standard-size range includes the reference-size scenario variants and the vendor-target stress-test scenarios.

Policy class	Standard-size scenarios	Larger network	More demand
R-D	0.22–1.68(Uniform: 15.27)	1.65	0.80
DCL	53.75–143.02	374.85	339.13

it requires a CF-LP evaluation during execution. This reinforces the interpretation that CF-LP is useful as a benchmark but less attractive as a live online policy.

Table 3.7: Online execution time per allocation step by policy class. Times are reported in milliseconds. For timing only, all policies except CF-LP were run over 10,000 trajectories; CF-LP was run over 1,000 trajectories. All performance comparisons in the main paper use the common 1,000-trajectory evaluation set.

Policy class	Standard-size & More demand	Larger network
M-MI	0.00046–0.00052	0.00067
M-S	0.00051–0.00063	0.00087
R-D	0.00048–0.00053	0.00063
CF-PD	0.00054–0.00065	0.00080
DCL	0.01913–0.01996	0.02956
CF-LP	11.32–11.63	11.71

Bibliography

Jason Acimovic and Vivek F. Farias. *The Fulfillment-Optimization Problem*, chapter 9, pages 218–237. INFORMS, 2019. doi: 10.1287/educ.2019.0199.

Jason Acimovic and Stephen C. Graves. Making better fulfillment decisions on the fly in an online retail environment. *Manufacturing & Service Operations Management*, 17(1): 34–51, 2015. doi: 10.1287/MSOM.2014.0505.

Niels A.H. Agatz, Moritz Fleischmann, and Jo A.E.E. van Nunen. E-fulfillment and multi-channel distribution - a review. *European Journal of Operational Research*, 187(2): 339–356, 2008. doi: 10.1016/j.ejor.2007.04.024.

Fabian Akkerman, Luca Begnardi, Riccardo Lo Bianco, Tarkan Temizoz, Martijn Mes, and Willem van Jaarsveld. Dynaplex, 2023. URL https://github.com/DynaPlex/DynaPlex. Software repository.

Alibaba Group Holding Limited. Fiscal year 2026 annual report. Annual report, Alibaba Group Holding Limited, 2026. URL https://www.sec.gov/Archives/edgar/data/1577552/000119312526231755/baba-20260331.htm. Accessed: 2026-08-18.

Ayoub Amil, Ali Makhdoumi, and Yehua Wei. Multi-item order fulfillment revisited: LP formulation and prophet inequality. *Management Science*, 71(12):9917–9935, 2025. doi: 10.1287/mnsc.2023.00357.

John M. Andrews, Vivek F. Farias, Aryan I. Khojandi, and Chad M. Yan. Primal–dual algorithms for order fulfillment at Urban Outfitters, Inc. *INFORMS Journal on Applied Analytics*, 49(5):355–370, 2019. doi: 10.1287/INTE.2019.1013.

Robert N. Boute, Joren Gijsbrechts, Willem van Jaarsveld, and Nathalie Vanvuchelen. Deep reinforcement learning for inventory control: A roadmap. *European Journal of Operational Research*, 298(2):401–412, 2022. doi: 10.1016/j.ejor.2021.07.016.

European Commission. Find the best price for your EU parcel delivery, 2025. European Commission parcel-delivery price-comparison data. Experiments use the 2025 tariffs for a 1kg track-and-trace parcel delivery service. The online page is updated annually and may display a later tariff year. Accessed February 10, 2026.

Bram J. De Moor, Joren Gijsbrechts, and Robert N. Boute. Reward shaping to improve the performance of deep reinforcement learning in perishable inventory management. *European Journal of Operational Research*, 301(2):535–545, 2022. doi: 10.1016/j.ejor.2021.10.045.

Eurostat. Population on 1 january, 2024. Dataset code: tps00001__custom_15826080. Population values for 2024 used to construct regional demand weights. Accessed March 17, 2025.

Alexander Hübner, Andreas Holzapfel, and Heinrich Kuhn. Operations management in multi-channel retailing: an exploratory study. *Operations Management Research*, 8(3-4):84–100, 2015. doi: 10.1007/S12063-015-0101-9.

Alexander Hübner, Heinrich Kuhn, and Johannes Wollenburg. Last mile fulfilment and distribution in omni-channel grocery retailing: A strategic planning framework. *International Journal of Retail and Distribution Management*, 44(3):228–247, 2016. doi: 10.1108/IJRDM-11-2014-0154.

Alexander Hübner, Jonas Hense, and Christian Dethlefs. The revival of retail stores via omnichannel operations: A literature review and research framework. *European Journal of Operational Research*, 302(3):799–818, 2022. doi: 10.1016/J.EJOR.2021.12.021.

Stefanus Jasin and Amitabh Sinha. An LP-based correlated rounding scheme for multi-item ecommerce order fulfillment. *Operations Research*, 63(6):1336–1351, 2015. doi: 10.1287/OPRE.2015.1441.

Guokai Li, Ningyuan Chen, Guillermo Gallego, Pin Gao, and Steven Kou. Dealership or marketplace with fulfillment services: A dynamic comparison. *Manufacturing & Service Operations Management*, 26(5):1860–1877, 2024a. doi: 10.1287/msom.2022.0253.

Jun Li, Wenjing Shen, Yi Liao, Gangshu Cai, and Xiangfeng Chen. The fulfillment service in online marketplaces. *European Journal of Operational Research*, 315(3):1139–1152, 2024b. doi: 10.1016/j.ejor.2024.01.003.

Jack Ma. Remarks at the economic club of new york. Transcript published by The Economic Club of New York, June 2015. URL https://www.econclubny.org/documents/10184/109144/2015MaTranscript.pdf. Accessed: 2026-07-29.

Will Ma. Order-optimal correlated rounding for fulfilling multi-item e-commerce orders. *Manufacturing & Service Operations Management*, 25(4):1324–1337, 2023. doi: 10.1287/msom.2023.1219.

Marco Melacini, Sara Perotti, Monica Rasini, and Elena Tappia. E-fulfilment and distribution in omni-channel retailing: a systematic literature review. *International Journal of Physical Distribution and Logistics Management*, 48(4):391–414, 2018. doi: 10.1108/IJPDLM-02-2017-0101.

Devesh Raval. Steering in one click: Platform self-preferencing in the amazon buy box. Unpublished manuscript, Harvard Law School, 2023. URL https://deveshraval.github.io/buyBox.pdf.

Jennifer K. Ryan, Daewon Sun, and Xuying Zhao. Competition and coordination in online marketplaces. *Production and Operations Management*, 21(6):997–1014, 2012. doi: 10.1111/j.1937-5956.2012.01332.x.

Huiyun Shen, Jie Yu, Hua Zhang, Jin Gou, and Xiangqian Zhang. How does social support affect the retention willingness of cross-border e-commerce sellers? *Frontiers in Psychology*, 12:797035, 2022. doi: 10.3389/fpsyg.2021.797035.

Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2219–2228. Association for Computing Machinery, 2018. doi: 10.1145/3219819.3220088.

Heshan Sun. Sellers' trust and continued use of online marketplaces. *Journal of the Association for Information Systems*, 11(4):182–211, 2010. doi: 10.17705/1jais.00226.

Libo Sun, Guodong Lyu, Yugang Yu, and Chung-Piaw Teo. Fulfillment by amazon versus fulfillment by seller: An interpretable risk-adjusted fulfillment model. *Naval Research Logistics*, 67(8):627–645, 2020. doi: 10.1002/nav.21897.

Josef Svoboda and Stefan Minner. A data-driven approach for strategic inventory placement in multi-echelon supply networks. *European Journal of Operational Research*, 328 (2):446–459, 2026. doi: 10.1016/j.ejor.2025.06.022.

TechNet. Technet study reveals valuable role of e-commerce marketplaces in supporting small businesses. TechNet, December 2024. URL https://www.technet.org/media/technet-study-reveals-valuable-role-of-e-commerce-marketplaces-in-supporting-small-businesses/. Accessed: 2026-07-02.

Tarkan Temizöz, Christina Imdahl, Remco Dijkman, Douniel Lamghari-Idrissi, and Willem van Jaarsveld. Deep controlled learning for inventory control. *European Journal of Operational Research*, 324(1):104–117, 2025. doi: 10.1016/j.ejor.2025.01.026.

Lin Tian, Asoo J. Vakharia, Yinliang Tan, and Yifan Xu. Marketplace, reseller, or hybrid: Strategic analysis of an emerging e-commerce model. *Production and Operations Management*, 27(8):1595–1610, 2018. doi: 10.1111/poms.12885.

Sérgio Vasconcelos, Gonçalo Figueira, and Bernardo Almada-Lobo. Order allocation in online retail: Classification and literature review. *European Journal of Operational Research*, 333(1):1–18, 2026. doi: 10.1016/j.ejor.2025.11.018.

Yihua Wang and Stefan Minner. Deep reinforcement learning for demand fulfillment in online retail. *International Journal of Production Economics*, 269:109133, 2024. doi: 10.1016/J.IJPE.2023.109133.

Yihua Wang and Stefan Minner. Data-driven multi-location inventory placement in digital commerce. *Computers & Industrial Engineering*, 200:110842, 2025. doi: 10.1016/j.cie.2024.110842.

Ping Josephine Xu, Russell Allgor, and Stephen C. Graves. Benefits of reevaluating real-time order fulfillment decisions. *Manufacturing & Service Operations Management*, 11 (2):340–355, 2009. doi: 10.1287/MSOM.1080.0222.

CHAPTER 4

Decision Support Framework for Sequential Order Allocation

Data-Driven Decision Support Framework for Sequential Order Allocation – a Case Study in a Global Online Fashion Marketplace

Sérgio Vasconcelos* · **Gonçalo Figueira*** · **Bernardo Almada-Lobo***

Manuscript submitted to *Computers & Industrial Engineering*, under review.

Abstract Order allocation in online retail assigns each incoming unit of demand to a fulfillment node in real time, under dynamic inventory, feasibility constraints, and uncertainty. Although optimization-based approaches offer strong theoretical guarantees, many rely on assumptions that are difficult to satisfy in practice, such as stationary demand and reliable forecasts. Retailers, therefore, tend to rely on robust and interpretable rule-based policies, yet have limited support for evaluating or improving them across operating scenarios.

We propose a simulation-based decision-support framework to evaluate and optimize fulfillment allocation policies in retail environments. The framework models order allocation as a sequential decision process, uses a discrete-event simulator to evaluate candidate policies under controlled scenarios, and supports interpretable scoring-based rules. Genetic programming instantiates the optimization layer, generating and refining rules from simulation feedback. Separate training, validation, and testing scenarios guide policy search, candidate screening, and out-of-sample assessment.

We apply the framework to a global online fashion marketplace using real operational data covering approximately 57,000 items and over 1 million orders. The case study shows that simulation-based policy optimization can operate at a realistic scale, generate compact allocation rules, and reduce fulfillment costs relative to current practice without degrading service-level performance. Scenario analyses further expose robustness trade-offs across inventory and cost conditions. The framework thus supports stress-testing and improving sequential order-allocation policies prior to deployment.

Keywords Online retail ·order allocation ·decision support ·simulation-optimization ·genetic programming

*INESC TEC, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal

4.1. Introduction

Order allocation in online retail determines which fulfillment node (e.g., a warehouse, store, or third-party vendor) should satisfy each incoming customer order. As retail networks become more fragmented and diversified, allocation decisions play a central role in controlling fulfillment costs, meeting delivery commitments, and maintaining service reliability. The problem is particularly challenging in large-scale e-commerce systems, where many orders, products, and potential fulfillment sources interact dynamically over time. Allocation policies must therefore balance inventory availability, fulfillment cost, delivery speed, and service-level requirements while remaining computationally efficient and operationally deployable.

Order allocation naturally unfolds as a sequential decision problem. Orders arrive over time, and due to often high demand and customer expectations of fast fulfillment, many large-scale settings require near-real-time, order-by-order decisions. Each allocation changes the future system state, most notably by consuming inventory at specific fulfillment nodes, and can therefore affect the feasibility and quality of subsequent decisions. At the same time, the environment is uncertain: future demand is unknown, inventory records may be imperfect or delayed, and operational performance can vary across fulfillment nodes and over time. These characteristics make purely myopic allocation rules and long-horizon deterministic optimization models difficult to rely on.

These challenges arise across several online retail configurations, including centralized e-commerce networks, omnichannel retailers, and marketplaces. Although these settings differ in ownership structure, fulfillment flexibility, inventory visibility, and operational control, they share a common decision-support requirement: allocation policies must make fast, feasible, and sequential decisions while balancing cost, service, and inventory implications. Additional operational characteristics, such as heterogeneous lead times, limited fulfillment flexibility, sparse demand, or item-specific handling requirements, further increase the need for policies that are robust, interpretable, and compatible with live fulfillment systems.

The academic literature offers a wide range of optimization and control approaches for order allocation, but many rely on assumptions that are difficult to satisfy in practice. Common assumptions include accurate and timely inventory information, stationary demand processes, homogeneous fulfillment nodes, and reliable demand forecasts (Acimovic and Graves, 2015; Jasin and Sinha, 2015). These assumptions simplify the analysis but can limit direct deployment in environments with sparse demand, volatile assortments, asynchronous inventory updates, or heterogeneous operational capabilities. As a result, a gap remains between theoretically motivated allocation policies and decision-support tools that can be readily implemented in real fulfillment systems.

In practice, many retailers rely on rule-based allocation policies. These policies are attractive because they are transparent, easy to communicate to stakeholders, and straightforward to integrate into existing systems. They can also be calibrated using historical data on allocations, inventory levels, and operational performance without requiring explicit forecasts. However, practical tools for systematically developing, comparing, and improving such policies remain limited. The practical adoption of integrated simulation-based

frameworks capable of reproducing realistic fulfillment dynamics, evaluating competing policies under controlled scenarios, and assessing policy behavior before deployment remains limited (Le and Fan, 2024).

To address this need, we propose a Decision-Support Framework (DSF) for sequential order allocation in online retail. The framework uses historical data to construct simulation instances in which interpretable, scoring-based allocation policies are evaluated under realistic inventory dynamics and operational uncertainty. Simulation supports controlled policy evaluation, while policy optimization, implemented using GP, enables the automated discovery of improved decision rules within a transparent and operationally meaningful policy class. In this way, the framework separates feasibility handling, which is enforced by the simulator, from preference learning, which is expressed through deployable scoring rules.

We instantiate the framework in a real-world online fashion marketplace, where the platform determines order allocation, while independent vendors retain control over their own inventory and fulfillment operations. Given the observed sparse-demand setting, we construct training, validation, and testing instances from distinct product sets, retaining their demand histories and using order co-occurrence to limit fragmentation of multi-item orders. This design is better suited to settings dominated by long-tail items, where temporal splits may yield unstable or uninformative evaluation samples. The case study ultimately illustrates how simulation-based decision support can be used to compare current-practice allocation rules, identify improved interpretable policies, and analyze robustness under alternative inventory and cost conditions. The paper makes three contributions to order allocation optimization for online retail fulfillment: (i) a modular framework that separates feasibility enforcement from policy scoring and uses a common simulator for incumbent-policy evaluation and new-policy search; (ii) a GP-based procedure that improves scoring rules built from operational features and screens them by fulfillment cost and expression complexity; and (iii) a real-world marketplace evaluation based on train–validation–test splits across distinct products, showing that compact rules can improve the operational benchmark while exposing cost–service trade-offs.

The remainder of the paper is organized as follows. Section 4.2 reviews related literature on order allocation, simulation-based optimization, and decision support. Section 4.3 defines the business problem and decision context. Section 4.4 presents the proposed simulation–optimization framework. Section 4.5 reports the case-study application, empirical results, and managerial implications. Section 4.6 concludes with limitations and future research directions.

4.2. Literature Review

Online retail fulfillment has motivated a substantial literature on online and order-by-order allocation policies, in which sequential decisions must be made under inventory and capacity constraints. Early work studied dynamic order allocation in a two-store network with drop-shipping (Xiao et al., 2009), while subsequent contributions developed opportunity-cost approximations based on linear programming (Acimovic and Graves, 2015) and primal-dual methods (Andrews et al., 2019). Other studies addressed multi-item

fulfillment and network structure, including randomized policies for shipment consolidation (Jasin and Sinha, 2015) and greedy policies for networks with limited fulfillment flexibility (Asadpour et al., 2019). For a detailed review of order allocation models in online retail and their solution methods, see Vasconcelos et al. (2026). These approaches provide important algorithmic insights and, in some cases, performance guarantees. However, they are typically designed to derive allocation policies under stylized assumptions, rather than to support the simulation-based evaluation, improvement, and stress-testing of deployable allocation rules under realistic operational conditions.

A related stream of work uses data-driven methods to support operational decision-making. Prescriptive analytics and decision-focused learning provide general frameworks for learning decisions from data (Bertsimas and Kallus, 2020; Bengio et al., 2021), while interpretable rule-based and scoring-based methods have been proposed to improve transparency and operational trust in data-driven decision systems (Bertsimas et al., 2019; Cai et al., 2023). In retail operations, data-driven approaches have also been applied to inventory placement (Wang and Minner, 2025) and replenishment decisions (Punia et al., 2020; Harsha et al., 2025). Within this broader context, GP is particularly relevant because it can automatically search over mathematical expressions that define heuristic or score-based decision rules. Prior applications in scheduling (Branke et al., 2016; Ferreira et al., 2022) and inventory control (Lopes et al., 2020; Ghaddar et al., 2016) illustrate its suitability for dynamic operational settings. Nevertheless, existing data-driven approaches give less attention to improving current-practice-compatible scoring rules for real-time fulfillment allocation, where interpretability, feasibility, and deployability are central requirements. Our approach addresses this gap by using GP to search directly within the scoring-rule paradigm used in practice, thereby improving policy performance without replacing the underlying decision logic with an opaque model.

Finally, this work is connected to the literature on ML–based DSFs, including applications in fashion online retail supply chains, where decision-making must account for evolving customer preferences and changing product availability (Pereira et al., 2022). Such frameworks increasingly combine prediction, optimization, and simulation to support operational decisions under uncertainty. Coelho et al. (2021) propose a simulation-based decision support tool for in-house distribution and production facilities, but their simulation is a virtual environment for system analysis and scenario evaluation. Our work couples the simulator with an algorithm that searches over operational decision policies and returns improved rules. In retail and omnichannel contexts, ML–enabled decision support has been used for fulfillment routing (Gallino et al., 2021), assortment and pricing (Cai et al., 2023), and inventory replenishment (Harsha et al., 2025). However, much of this literature focuses on embedding predictive models within optimization or control procedures, or on supporting adjacent retail decisions. Our contribution is to integrate historical operational data, offline policy evaluation, and ML-based improvement of interpretable allocation rules into a common workflow in which the same simulator supports incumbent-policy evaluation, automated policy search, validation, testing, and robustness analysis before deployment.

4.3. Business Problem and Decision Context

4.3.1 Problem Description and Decision Model

We consider an order allocation problem arising in online retail fulfillment, where customer demand arrives sequentially in the form of orders. Each order may contain demand for one or multiple units of one or multiple items, and these units must be allocated to fulfillment nodes in near real time. Upon order arrival, allocation decisions are made before future demand is observed. Hence, decisions are sequential and irrevocable: allocating inventory to the current order changes the inventory available for subsequent orders and may affect future fulfillment performance.

Let t denote an order-arrival epoch, and let o_t be the set of demand units in the order arriving at that epoch. Each unit $u \in o_t$ is associated with an item $i(u) \in I$. Let J denote the set of physical fulfillment nodes and let j^0 denote an unfulfilled-demand option. $F_{u,t} \subseteq J$ denotes the set of physical nodes that are eligible to fulfill unit u at epoch t, according to operational and business feasibility rules. The binary variable $x_{j,u,t}$ equals 1 if unit u arriving at epoch t is assigned to option $j \in F_{u,t} \cup \{j^0\}$, and 0 otherwise. For each physical node $j \in J$, $s_{i,j,t}$ denotes the available inventory of item i immediately before the allocation decision at epoch t.

The decision-maker seeks an allocation policy $\pi \in \Pi$ that maps the observable system state at each epoch to feasible allocation decisions. The objective is to maximize expected cumulative reward over the planning horizon τ:

$$\max_{\pi \in \Pi} \ \mathbb{E}^{\pi}\left[\sum_{t=1}^{\tau} R_t\right] \tag{4.1}$$

$$\text{s.t.} \sum_{j \in F_{u,t} \cup \{j^0\}} x_{j,u,t} = 1, \qquad \forall u \in o_t,\ t, \tag{4.2}$$

$$\sum_{u \in o_t : i(u)=i,\ j \in F_{u,t}} x_{j,u,t} \le s_{i,j,t}, \qquad \forall i \in I,\ j \in J,\ t, \tag{4.3}$$

$$x_{j,u,t} \in \{0,1\}, \qquad \forall j \in F'_{u,t},\ u \in o_t,\ t. \tag{4.4}$$

Equation (4.1) defines the policy objective, where R_t represents the reward collected at epoch t. Depending on the application, this reward may encode a single performance metric, such as negative shipping cost, or a weighted combination of operational objectives, such as cost, service level, and unfulfilled demand. Constraints (4.2) ensure that each demand unit is assigned to exactly one feasible allocation option, including the unfulfilled-demand option when physical fulfillment is infeasible or undesirable. Constraints (4.3) enforce local inventory feasibility at each decision epoch. Constraints (4.4) define the binary nature of unit-level allocation decisions.

Inventory evolves dynamically as a result of allocation decisions and exogenous inventory movements. Allocating a unit to a physical node depletes the corresponding inventory, while other changes may occur independently of the allocation policy. These exogenous movements can represent replenishments, returns, external sales, or stock corrections received asynchronously from fulfillment partners. Similarly, eligibility rules can

encode vendor availability, service-region restrictions, fulfillment-mode eligibility, carrier constraints, or item-specific requirements. The model therefore remains compact while allowing the decision environment to reflect operational realities commonly observed in online retail fulfillment.

Assumptions followed in our model are that (1) decisions are made immediately after order arrival, (2) all order information required for allocation is observed at the decision epoch, (3) inventory information is available to the decision system, and (4) cross-order interactions arise primarily through shared inventory, since fulfilling earlier orders consumes inventory that may otherwise have served later orders. These assumptions are consistent with direct-shipping settings in which order allocation policies must make fast, feasible, and operationally compliant decisions.

A detailed formulation of the sequential decision process, including the state variables, exogenous information, transition function, reward structure, and induced optimization problem, is provided in Appendix 4.A.

4.3.2 Business Motivation and Use Cases

The DSF proposed in this paper is designed to support the evaluation and optimization of allocation policies for the decision problem described above. It is motivated by operational needs that arise in online retail environments characterized by sequential decision-making, heterogeneous fulfillment options, complex business rules, and abundant operational data. These needs were identified through interactions with a large online fashion marketplace, but they are shared with other online retail settings where fulfillment decisions must remain fast, transparent, and compatible with production systems.

Use case 1: Policy evaluation. In many online retail organizations, fulfillment policies are manually specified as heuristics, priority rules, or combinations of hard constraints and ranking criteria. These policies are designed to reflect business objectives, such as reducing fulfillment cost, protecting service levels, or prioritizing specific fulfillment options, but their behavior is often assessed only after deployment. This makes it difficult to understand how a policy would perform under alternative demand patterns, inventory configurations, cost conditions, or strategic priorities.

Simulation provides a way to evaluate current and candidate policies before deployment. By replaying or generating realistic operational scenarios, decision-makers can compare policies under controlled conditions, quantify trade-offs among cost, service, and unfulfilled demand, and identify overly conservative or fragile policies. This is particularly useful when hard constraints are used to avoid undesirable outcomes, but may also exclude economically attractive fulfillment decisions. In such settings, simulation-based evaluation reduces deployment risk and supports a more informed policy selection.

Use case 2: Policy optimization. Many online retailers maintain historical records of orders, inventory movements, allocation decisions, and downstream performance outcomes. Although these data are often used for reporting or forecasting, they can also support the systematic improvement of allocation policies. In the motivating case study, and in many

operational settings, allocation logic can be expressed through scoring or priority rules: candidate fulfillment nodes are scored using observable attributes, and the selected node is the feasible option with the most favorable score. Such policies are practically attractive because they are interpretable, auditable, and straightforward to implement.

This motivates policy optimization methods that operate directly over scoring-based decision rules. These practical scoring-based policies can be improved using simulation-based optimization methods, such as the Genetic Programming approach adopted in this paper. It preserves the underlying decision paradigm by using historical operational data to generate, evaluate, and refine compatible yet higher-performing scoring rules in a simulated environment before testing them in live systems.

Together, the use cases define the requirements for the proposed DSF, which must reproduce realistic fulfillment dynamics, evaluate existing and candidate policies under controlled scenarios, and support offline optimization of interpretable allocation rules prior to deployment.

4.4. Decision Support Framework for Policy Evaluation and Optimization

This section proposes a DSF that transforms historical operational data and business rules into simulation instances. These instances feed into a simulation engine that supports both policy evaluation and policy optimization, enabling existing or manually designed policies to be compared under controlled scenarios, while GP can use repeated simulation feedback to search for improved scoring-based allocation rules.

4.4.1 Overview and System Architecture

Figure 4.1 summarizes the proposed architecture. The region above the horizontal dashed line represents the online fulfillment environment, where orders are allocated, policies are deployed, and live performance is monitored. The remaining figure represents the offline analytics environment, where historical data are transformed into simulation instances, candidate policies are evaluated, and optimized policies are generated before deployment.

The online layer generates the operational data required by the decision-support process. These data may include historical orders, item and node information, inventory snapshots or updates, allocation decisions, eligibility rules, and fulfillment cost or service information. In the offline layer, a data-processing module converts these records into simulation instances that reproduce the relevant decision context. This preparation is application-dependent. For example, it may require reconstructing historical inventory, defining feasible node sets, estimating cost matrices, or constructing counterfactual scenarios for policy comparison.

The framework supports the two main use cases introduced in Section 4.3.2. In policy evaluation, business managers provide one or more current or candidate allocation policies, which are then evaluated on testing instances through the simulator. Such an approach supports controlled comparison of incumbent policies, manually designed rules, and policies

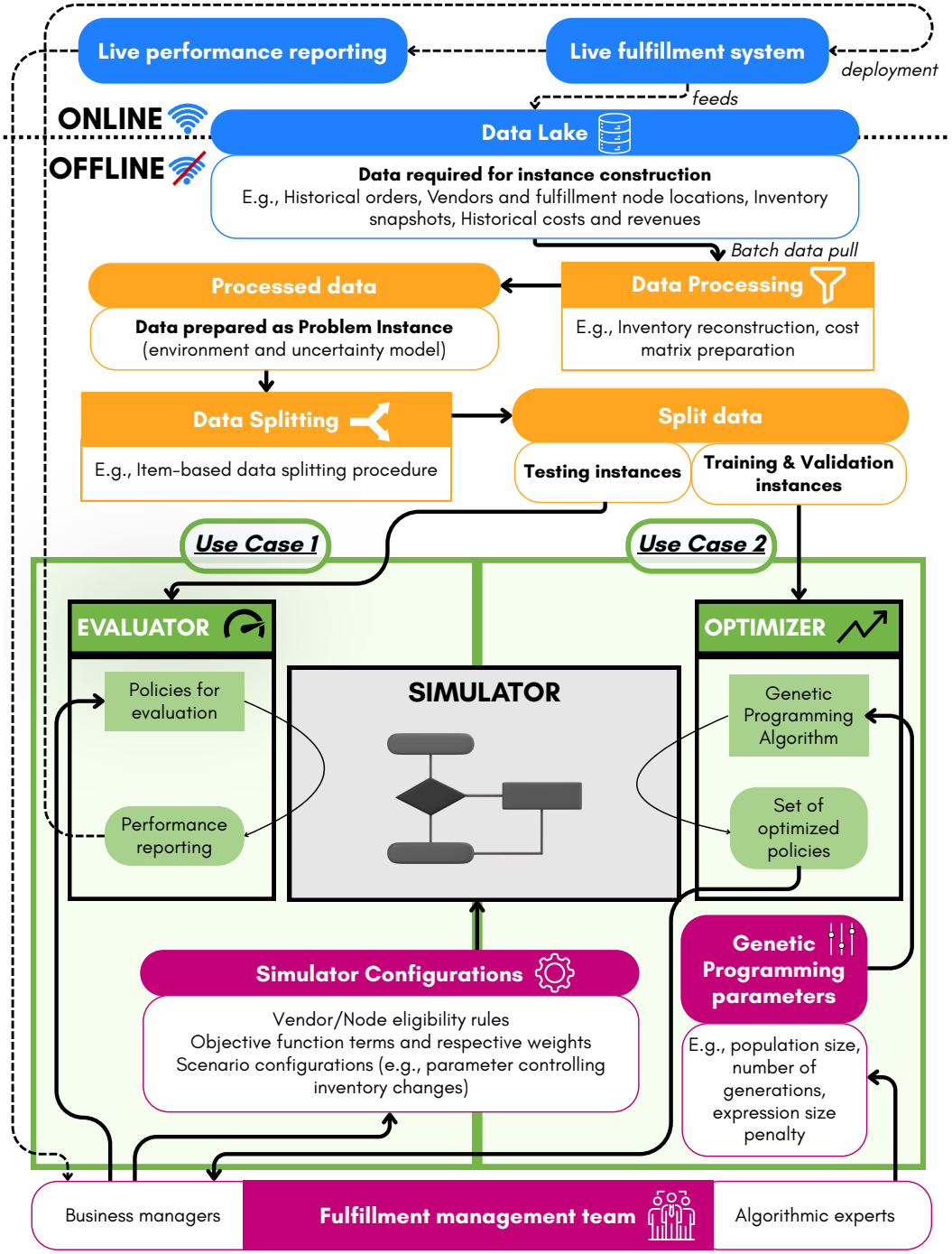


Figure 4.1: System architecture for simulation-based policy evaluation and optimization.

obtained through optimization. In policy optimization, GP uses training instances to generate and refine scoring-based candidate policies. The resulting policies are then screened on validation instances, and selected candidates can subsequently be assessed through the policy-evaluation workflow on testing instances.

The architecture also distinguishes the roles of data, simulation, optimization, and managerial intervention. The simulator is the common execution engine that receives a simulation instance and a policy, enforces feasibility, simulates the allocation process, and returns performance metrics. The optimizer does not require direct access to the internal state-transition logic; it only requires candidate-policy evaluations returned by the simulator. This modularity allows the same simulator to be used for incumbent-policy assessment, policy search, validation, testing, and robustness analysis under alternative demand, inventory, or cost scenarios.

Overall, the architecture creates a controlled offline environment for improving allocation decisions before live deployment. Historical data provide realistic empirical grounding, simulation enables counterfactual policy evaluation, and GP uses that same simulation environment to improve interpretable scoring rules while preserving a deployable decision paradigm. In the remainder of this section some specific components of this framework are analyzed in more detail: we define the scoring-based policy class, describe the simulation engine used to evaluate such policies, and explain how GP uses simulation feedback for policy optimization.

4.4.2 Scoring-Based Allocation Policies

The framework evaluates and optimizes policies that can be expressed as scoring-based allocation rules. This policy class is practically relevant because many operational allocation systems rank feasible fulfillment options using cost indices, priority rules, or weighted combinations of observable attributes.

At each order-arrival epoch t, for each demand unit $u \in o_t$, the simulator constructs the feasible set $F_{u,t}$ of physical fulfillment nodes satisfying the current inventory and eligibility requirements. A scoring-based policy assigns a scalar score to each feasible option:

$$\theta(j,u,t), \qquad j \in F_{u,t} \cup j^0,$$

where θ denotes the scoring expression, and $F_{u,t} \cup j^0$ the set of feasible physical nodes plus the unfulfilled option, consistent with Constraint (4.2). The selected option is then

$$j^*(u,t) = \underset{j \in F_{u,t} \cup j^0}{\arg\max}\, \theta(j,u,t),$$

with the corresponding assignment variable set to $x_{j^*(u,t),u,t} = 1$. If the implementation uses lower scores as preferable, the same representation can be written equivalently as an arg min rule.

The scoring function may combine observable features, constants, and operators. Features can describe, for example, fulfillment cost, inventory availability, inventory scarcity (e.g., stock depth), lead-time proxies, node attributes, order attributes, or item-specific char-

acteristics (e.g., weight). Operators define how these features are combined into a single scalar score. They can be arithmetic (e.g., addition, multiplication) or other mathematical operators (e.g., square root, logarithm). This representation is flexible enough to encode manually designed allocation rules while also being suitable for automated search.

A key modeling choice is that the policy ranks only feasible options. Feasibility is not learned by the policy; it is enforced by the simulator through the construction of $F_{u,t}$ and through the inventory and assignment requirements in Constraints (4.2)–(4.4). This separation keeps the learned or optimized policy focused on preference ordering among admissible choices, while preserving operational validity by construction.

4.4.3 Simulation Engine

The simulator operationalizes the decision model introduced in Section 4.3.1. It evaluates a given policy by executing a discrete-event simulation over a sequence of order arrivals. At each epoch, the simulator observes the arriving order, constructs feasible allocation options, invokes the policy, updates inventory, computes rewards, and records performance metrics. The simulation is used to report policy performance and serves as the core evaluation mechanism by the optimizer.

A simulation instance distinguishes between the *structural environment* and the *scenario realization*. The structural environment defines the elements and rules that remain fixed across a set of simulations: the item set, fulfillment nodes, eligibility logic, policy features, reward components, and the baseline mapping from allocation decisions to cost or service outcomes. The scenario realization specifies the time-dependent inputs used in a particular simulation run, including the initial inventory state, order sequence, exogenous inventory movements, and any scenario-specific changes to rewards or costs. This distinction is useful for robustness analysis. For example, the same structural environment can be evaluated under alternative inventory realizations to assess sensitivity to stock availability, or under modified cost realizations to test whether a policy remains effective when shipping-cost conditions change.

Algorithm 4 summarizes the execution of a simulation run for a given scoring-based policy θ. The simulator enforces the assignment and feasibility requirements in Constraints (4.2)–(4.4). The realized rewards are accumulated to estimate the objective in Equation (4.1).

The feasible set $F_{u,t}$ is reconstructed at each unit-allocation step because prior allocations within the same order, or earlier orders in the trajectory, may have changed inventory availability. If $F_{u,t}$ is empty, only the unfulfilled-demand option remains available. If multiple physical nodes are feasible, the policy ranks them together with the unfulfilled-demand option, depending on how the latter is represented in the scoring rule. This mechanism ensures that unit-level decisions remain feasible while still allowing the policy to express preferences over cost, service, inventory preservation, or other operational criteria.

The simulator records performance at different aggregation levels. Unit-level rewards are computed at allocation time and can be aggregated into order-level metrics R_t that summarize an order's outcome after all its units have been allocated (e.g., the order's fulfillment cost). Run-level metrics aggregate performance over the full horizon (e.g., the horizon's to-

Algorithm 4 Simulation run for a given allocation policy

Require: Simulation instance; policy θ
1: Initialize inventories $s_{i,j,1}$ and performance metrics
2: **for** $t = 1, \ldots, \tau$ **do**
3: Observe arriving order o_t & initialize epoch reward $R_t \leftarrow 0$
4: **for** each demand unit $u \in o_t$ **do**
5: Construct feasible set $F_{u,t}$ using inventory and eligibility rules
6: Select $j^*(u,t) = \arg\max_{j \in F_{u,t} \cup j^0} \theta(j,u,t)$
7: Set $x_{j^*(u,t),u,t} = 1$
8: Update inventory for item $i(u)$ at node $j^*(u,t)$
9: Compute the unit-level reward contribution and update R_t
10: **end for**
11: Record order-level performance metrics
12: Apply exogenous inventory movements before the next epoch
13: **end for**
14: **return** run-level performance metrics

tal fulfillment cost). These outputs support policy reporting, policy comparison, robustness analysis, and the fitness calculations used by GP. Because optimization requires many repeated evaluations, the simulator must support efficient state updates, low-overhead policy execution, and consistent metric aggregation across scenarios.

4.4.4 GP-Based Policy Optimization

The optimizer uses GP to search over scoring-based allocation policies. Each GP individual encodes a symbolic scoring expression built from a predefined terminal set, such as operational features and constants, and a function set, such as mathematical operators. Once decoded as a policy π_θ, the individual can be evaluated by the simulation engine described in Algorithm 4. Policy quality is therefore measured through simulated fulfillment performance.

The optimization process follows a training–validation–testing pipeline. Training instances guide the evolutionary search by assigning fitness values to candidate policies. Validation instances provide out-of-sample screening of generated policies and support comparison across GP configurations. Finally, testing instances are reserved for the final assessment of selected policies and are used to produce performance reports for managerial interpretation.

Algorithm 5 summarizes the GP-based optimization workflow. The algorithm explicitly relies on repeated calls to the simulation run in Algorithm 4.

During training (Lines 1–9), GP evolves a population of scoring expressions. Each expression θ induces a scoring-based allocation policy and is evaluated on one or more training instances using the simulation engine. The resulting simulation outputs are aggregated into a scalar fitness value $F(\theta)$. The fitness function may correspond directly to the cumulative reward in Equation (4.1), reflecting fulfillment costs, unfulfilled demand, and/or service-level indicators, or it may include additional terms such as expression-size

Algorithm 5 GP-based optimization of scoring-based allocation policies

Require: Training, validation, and testing simulation instances; GP configuration
1: Initialize population $\mathcal{P}^0$ of scoring expressions
2: **for** generation $g = 1, \ldots, G$ **do**
3: **for** each scoring expression $\theta \in \mathcal{P}^{g-1}$ **do**
4: Evaluate the policy induced by θ on training instances using Algorithm 4
5: Assign training fitness $F(\theta)$ from aggregated simulation performance
6: **end for**
7: Select parent expressions according to fitness
8: Generate offspring using variation operators (e.g., crossover and mutation)
9: Form next population $\mathcal{P}^g = \text{Replacement}(\mathcal{P}^{g-1} \cup \text{Offsprings}^g)$
10: **end for**
11: Construct candidate set C from the generated scoring expressions
12: Evaluate candidate policies in C on validation instances using Algorithm 4
13: Select final candidate policies according to validation performance and complexity
14: Evaluate final candidate policies on testing instances using Algorithm 4
15: **return** selected policies and out-of-sample performance metrics

penalties, which are often used in GP to favor simpler and more interpretable rules. The GP configuration determines how the policy space is explored during the training loop. Relevant choices include the terminal and function sets, population size, initialization method, parent-selection rule (determines which expressions are more likely to generate offspring), crossover and mutation operators (create new candidate scoring rules), replacement strategy (determines how the next generation is formed), stopping criterion, and complexity-control mechanism (promotes the evolution of less complex rules, e.g., by penalizing larger expressions). These choices affect both search effectiveness and the interpretability of the resulting expressions.

After training, Line 11 constructs the candidate set C to be screened on validation instances. The definition of C depends on the optimization design: it may include all unique policies generated during training, only the final population, or a filtered archive of promising policies. Validation (Lines 12–13) is used as a relatively inexpensive screening procedure to evaluate a large set of policies generated during training and reduce overfitting to the training instances. Candidate policies are evaluated on separate validation instances using the same simulation engine.

Testing (Line 14) provides the final out-of-sample assessment of the selected policies. Its role is similar to validation in that it evaluates policies on instances not used during training, but it is applied only to the final selected policies and can therefore use more comprehensive, demanding, or computationally expensive evaluation conditions. The testing metrics may also differ from both the training fitness function and the validation metrics, depending on the managerial objectives of the final assessment. The resulting performance measures are used for policy comparison and managerial analysis. This final stage is particularly important in decision-support settings because the objective is to identify rules that are robust, interpretable, and credible enough to be considered for deployment.

Because GP interacts with the allocation problem only through the simulation interface (Algorithm 4), the optimization logic remains decoupled from the internal details of the simulator. Changes in eligibility rules, inventory dynamics, objective components, or scenario definitions can therefore be incorporated in the simulation model without changing the basic optimization workflow.

4.5. Application to Case Study

We apply the proposed DSF to a real-world online fashion marketplace fulfillment setting. The case study illustrates how historical orders, inventory records, shipping-cost proxies, and business rules can be transformed into simulation instances for policy evaluation and optimization. It also provides insight into whether interpretable scoring-based policies, optimized offline through simulation feedback, can improve over current allocation practice while remaining operationally deployable.

4.5.1 Case Study Context Description

The case study is based on a large online fashion marketplace operating a globally distributed fulfillment network. In this setting, the marketplace platform controls order allocation, while independent vendors manage their own inventory and fulfillment operations. This creates a challenging decision environment: allocation decisions must be made sequentially and at scale, but inventory availability, fulfillment capacity, and operational performance depend on distributed vendor-controlled processes.

Operational scale and complexity. The dataset spans approximately six months of operations and reflects a fulfillment environment of substantial scale:

- approximately 91,000 distinct items, each stocked at a subset of fulfillment nodes*;
- nearly 3,500 fulfillment nodes distributed across more than 250 geographic locations;
- more than 14,000 distinct shipping routes, defined by origin–destination country pairs.

The initial data scope therefore captures a large and heterogeneous fulfillment network, while focusing on products for which repeated and non-trivial allocation decisions can be observed.

Objectives of the case study. The case study pursues two objectives. First, it demonstrates that the proposed framework can be instantiated at a realistic scale using operational data and existing business rules. Second, it evaluates whether automatically generated scoring-based policies can improve over a manually designed operational rule while preserving interpretability, feasibility, and compatibility with current decision processes.

*The dataset made available for the study was limited to items stocked at multiple vendors and appearing in at least 10 orders over the observation horizon. These items account for 56% of the total marketplace order volume and provide an appropriate setting for this study, in which allocation choices are non-trivial and repeated over time.

Current operational policy. To anchor the study in current practice, we use the scoring features and baseline rule employed by the marketplace. The operational rule uses three features: Shipping Cost (SC), Stock Depth (SD), and Basket Coverage (BC). The SC feature is an approximate shipping-cost proxy, while SD is the current available inventory level of the item at the node. The BC feature is a within-order consolidation score: it takes value 0 if the node has already been assigned to a previous unit in the same order (meaning that this allocation will not result in additional costs), and otherwise takes value $1/(1+\mathit{OtherUnits})$, where *OtherUnits* is the number of remaining to-be-allocated units in the order that the node can fulfill (assuming that all those units will be allocated to this node, the cost is divided among them). In this case's implementation, we assume lower scores are preferable. The baseline rule is operationalized as

$$10000 \times BC + SC - \frac{SD}{10000}.$$

The scaling separates the components by order of magnitude relative to their observed ranges, thereby reproducing the hierarchical logic used in practice: prioritize consolidation, then shipping cost, and finally use inventory availability as a tie-breaker. This rule provides the benchmark against which learned policies are evaluated and motivates the terminal features used in the optimization setup.

4.5.2 Data and Instance Construction

The available historical data include order-level and item-level demand, order composition, delivery country, fulfillment-node locations, end-of-day item-level inventory snapshots, observed allocation decisions, and approximate shipping costs defined at the origin-country to destination-country level. Using observed allocations and inventory snapshots, we reconstructed daily exogenous inventory movements for each item–node pair. These movements represent inventory changes not explicitly modeled by the fulfillment policy, such as replenishments, withdrawals, corrections, or other vendor-side updates.

Inventory availability for counterfactual simulation was reconstructed by adding back historical allocations. This allows the simulator to evaluate alternative allocation policies while preserving the observed inventory dynamics. When replaying the operational policy described above, approximately 0.3% of demand could not be fulfilled. This discrepancy reflects the fact that the operational rule represents the dominant allocation logic but is not applied exclusively: alternative rules, exceptional cases, and manual overrides may occur in production. Given that the mismatch is small, no additional inventory adjustments were introduced.

Before constructing training, validation, and testing instances, the dataset was filtered to focus on cases where allocation decisions are meaningful and sufficiently supported by data. Items with an inventory-to-sales ratio greater than 1.25 were removed, leaving 72.8% of the original item set. Item–node pairs for which inventory was never available at the time of order arrival were then removed. Finally, items available at only a single node were excluded, leaving 62.6% of the original item set. These filters remove cases where allocation choice is either unlikely to affect performance or cannot be evaluated reliably.

After filtering, the dataset contained 57,666 items and approximately 1.15 million orders prior to splitting.

Rather than splitting the data temporally, we construct training, validation, and testing instances by partitioning items. This choice is motivated by the low-demand structure of the data: many products have sparse demand histories, so temporal splits would often produce short or uninformative decision sequences for individual items. Item-based splitting evaluates whether policies generalize across distinct product sets while preserving sufficiently rich demand trajectories within each instance. The procedure assigns approximately 5% of items to training, 5% to validation, and 90% to testing, yielding 2,884 training items, 2,883 validation items, and 51,899 testing items. The resulting instances contain 94,503, 97,598, and 1,009,638 orders, respectively.

Let $O(i)$ denote the set of orders containing item i, and let $O(I^k)$ denote the set of orders containing at least one item currently assigned to split $k \in \{\text{train}, \text{val}, \text{test}\}$. Algorithm 6 summarizes the item-based construction procedure.

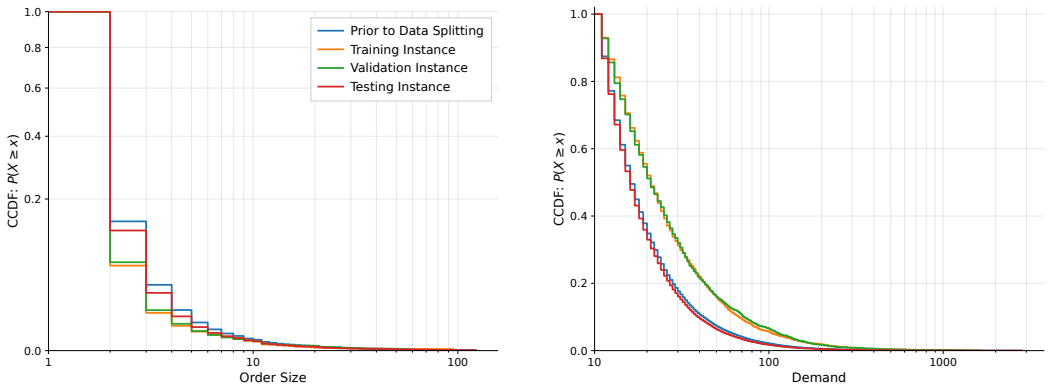
Algorithm 6 Item-based construction of training, validation, and testing instances

Require: Item set I, order data, target train-val-test proportions, balance threshold η

Construct the set of orders associated with each item i
repeat
    Initialize empty item sets I^{train}, I^{val}, and I^{test}
    Randomly order the items and seed each split with one item
    for each remaining item i **do**
        Compute $J(i,k) = \frac{|O(i) \cap O(I^k)|}{|O(i) \cup O(I^k)|}$ for each split k
        Assign item i to an eligible split using $\arg\max_k J(i,k)$ and the target proportions
    end for
    Construct split-specific order sequences by retaining demand units for assigned items (may require order splitting)
    Compute order-size and item-demand distribution differences across splits
until the maximum pairwise distributional difference is at most η
return Training, validation, and testing simulation instances

The overlap criterion $J(i,k)$ is based on order co-occurrence: items that frequently appear in the same orders are encouraged to remain in the same instance, reducing fragmentation of multi-item order structure. In the case study, the distributional balance threshold η was set to 7%, which was a reasonable level observed in experiments with the dataset. The resulting instances were assessed using the empirical distributions of order size and item demand, which are shown in Figure 4.2. The training and validation instances contain slightly fewer large orders than the unsplit dataset and the testing instance. For item demand, the training and validation curves show somewhat heavier low-to-moderate demand tails. These differences are expected because training and validation each contain only 5% of the items, making exact distributional matching difficult. Importantly, the testing instance remains close to the filtered dataset, supporting its use as a final out-of-sample assessment under conditions representative of the empirical setting.

At the end of this data-splitting process, each split induces a simulation instance con-



(a) Order size **(b)** Item demand

Figure 4.2: Empirical complementary cumulative distribution functions for the item-based data split: (a) order size, using a logarithmic x-axis and square-root-scaled y-axis; and (b) item demand, using a logarithmic x-axis.

taining the split-specific item set, associated order sequence, reconstructed initial inventory, exogenous inventory movements, eligibility information, and cost inputs.

4.5.3 Simulator Setup

The case-study simulator instantiates the simulation engine described in Section 4.4.3. It replays historical demand and reconstructed exogenous inventory movements deterministically. Thus, the order sequence, initial inventory, and external inventory updates are fixed within each simulation instance. Differences in simulated outcomes arise from the interaction between the evaluated policy and the evolving inventory state.

Exogenous inventory movements are not adapted in response to allocation decisions. In practice, vendor replenishment, rebalancing, or marketplace participation decisions may react to sustained changes in allocation behavior. Modeling such feedback would require an additional inventory-response layer. In this case study, fixing exogenous movements isolates the effect of allocation policy choice while retaining a dynamic and nonstationary inventory environment.

In this application, the simulator only enforces inventory availability when constructing feasible fulfillment options. Additional business constraints, such as vendor prioritization rules, contractual restrictions, or manual override logic, are not explicitly modeled in the empirical instantiation. This is a case-study simplification to make the results more interpretable; nevertheless, those additional constraints can be easily incorporated into the simulator.

The performance signal used in the case study is an approximate shipping cost defined at the origin-country to destination-country level (no additional granularity is available in the data). To represent heterogeneity among fulfillment nodes within the same origin country, controlled node-level variability is added to this matrix, yielding a fixed node–destination cost used whenever an order unit is allocated to a node. This cost proxy does not capture all relevant operational outcomes, such as returns, delivery delays, or environ-

mental impact. Nevertheless, it provides an interpretable measure of fulfillment efficiency and is sufficient to evaluate policy trade-offs in the available data.

4.5.4 Optimization Setup

This subsection specifies how the policy optimization component of the DSF is instantiated for the case study. The setup builds on the operational features introduced in Section 4.5.1, the scoring-policy representation introduced in Section 4.4.2, and the training–validation–testing instances constructed in Section 4.5.2. The main choices are summarized in Table 4.1.

All candidate policies, whether manually designed or automatically generated, are represented as scoring-based rules whose scores are minimized. The terminal set consists of the three operational features used by the baseline policy: BC, SC, and SD. The function set includes arithmetic and nonlinear operators implemented in protected form to ensure that every generated expression is executable within the simulator. No free numerical constants are included. This restricted representation prioritizes interpretability, generalizability, and alignment with existing business logic, ensuring that any generated policy is easy to operationalize.

The main GP configuration choices are reported in Table 4.1[†]. To assess robustness to stochasticity in the evolutionary process, each configuration is evaluated using 10 independent random seeds. Two crossover–mutation configurations are considered, and two fitness variants are used: one with an explicit size-based penalty and one without. Overall, this produces $200 \times 50 \times 2 \times 2 \times 10 = 400{,}000$ generated policy expressions.

During training, each policy is evaluated on the training instance through simulation. The primary performance signal is the average simulated fulfillment cost induced by the policy. In the penalized variant, this cost is combined with an expression-size penalty that increases with both expression complexity and generation index:

$$\text{Fitness}(\pi) = \frac{1}{\tau} \sum_{t=1}^{\tau} C_t(\pi) + \lambda \times \text{Size}(\pi) \times g. \tag{4.5}$$

Here, $C_t(\pi)$ is the fulfillment cost at epoch t under policy π, $\text{Size}(\pi)$ is the number of nodes in the policy expression tree, g is the generation index, and λ is the penalty weight. In the experiments, λ is set either to 0 or to 0.0002; the latter is small enough not to dominate the fulfillment-cost signal while biasing the search toward simpler policies.

After training, all syntactically unique generated policies are retained for validation. Among the 400,000 generated expressions, 206,701 syntactically distinct expressions were identified. Two expressions are treated as distinct when their symbolic representations differ, even if they are algebraically equivalent due to commutativity or redundant structure. This conservative uniqueness criterion preserves policy diversity without requiring expression simplification or canonicalization.

Remaining policies are evaluated on the validation instance using simulated fulfillment

[†]Other GP configuration choices are set as Koza's standard parameters in the ECJ Evolutionary Computation Toolkit (Luke, 1998).

Table 4.1: Case-study instantiation of the policy optimization procedure.

Component	Specification
Terminal set	BC, SC, and SD.
Function set	$+$, $-$, $\times$, protected $\div$, protected log, protected $\sqrt{}$, protected exp, negation, and inversion.
GP configuration	Population size: 200; # generations: 50; tournament selection of size 2; crossover–mutation probabilities: $(0.8, 0.2)$ and $(0.2, 0.8)$; 10 random seeds per configuration.
Training fitness	Average simulated fulfillment cost on the training instance, optionally combined with a size- and generation-based penalty – see Equation (4.5).
Validation candidates	Syntactically unique policies generated during training.
Validation and testing metrics	Simulated fulfillment cost and unfulfilled-demand rate on the corresponding instance.
Selection for testing	Subset of non-dominated policies based on validation fulfillment cost and expression size, as detailed in Section 4.5.5.2.

cost, unfulfilled-demand rate, and expression size[‡]. The transition from validation to testing is more selective. Candidate policies are screened using validation cost and expression size, and a subset of non-dominated policies is selected for testing. Testing then provides the final out-of-sample assessment using a significantly larger instance.

4.5.5 Results and Discussion

This section reports the empirical results of the case study. We first use the simulator to stress-test the incumbent policy under inventory perturbations. We then analyze the GP optimization process and validation-based policy screening. Finally, we evaluate the selected policies on the nominal testing instance and under inventory and cost perturbations.

4.5.5.1 Policy Evaluation Under Inventory Perturbations

We first illustrate the policy-evaluation use case by stress-testing the incumbent policy under alternative inventory-availability scenarios. Inventory is a central driver of the order-allocation problem because it determines the range of allocation options. We therefore simulate the baseline policy on the testing instance after applying perturbations from -50% to $+50\%$, in steps of 10%, to the available inventory of each item at each fulfillment node. Figure 4.3 reports the relative change in total fulfillment cost compared with the nominal case and the corresponding unfulfillment rate.

The results show asymmetric behavior. When inventory availability increases, total fulfillment cost decreases steadily, reaching a reduction of 3.8% in the $1.5\times$ scenario. This

[‡]The unfulfilled-demand rate is nearly constant across policies under the reconstructed nominal inventory levels, hence why it is not included in the training fitness. Nevertheless, it becomes more relevant in the inventory-perturbation analyses reported later.

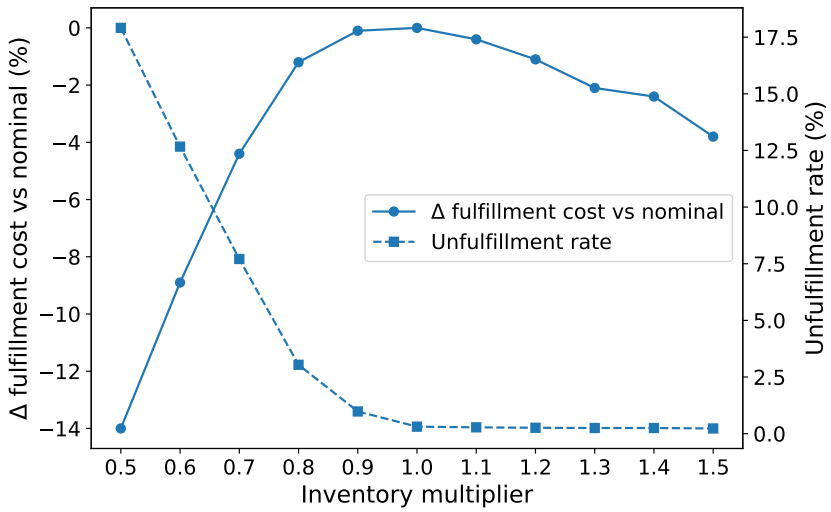


Figure 4.3: Sensitivity of the baseline policy to inventory availability on the testing instance.

reflects the value of additional feasible fulfillment options. Conversely, when inventory availability decreases, the unfulfillment rate rises sharply, reaching 17.9% in the 0.5× scenario (total fulfillment cost decreases because of fewer units being fulfilled). The experiment, therefore, shows how simulation-based stress testing is useful to determine the value of additional inventory, an important insight when motivating vendors to commit more stock to the marketplace.

4.5.5.2 Policy Optimization and Validation Screening

The policy used so far can be improved with the policy optimization module. We now analyze whether GP generates progressively better scoring-based policies and how validation is used to screen the resulting expressions.

Figure 4.4 reports validation cost versus expression size and highlights the non-dominated front. A policy is considered non-dominated if no other policy has both equal or smaller expression size and lower validation cost. The results show that most of the performance improvement can be achieved with compact expressions. In particular, the overall best expression has size 118, but its validation cost is less than 0.01 percentage points below the best policy of size 12. Given this negligible improvement relative to the additional complexity, the remaining analysis focuses on policies of size 12 or smaller.

Five policies are selected from representative complexity ranges on the validation frontier: expression sizes 1–2, 3–5, 6, 7–11, and 12. These policies correspond to P1–P5 in Table 4.2. P1 minimizes SC alone; P2 minimizes $SC \times BC$; P3 and P4 add inventory-awareness through logarithmic transformations of SD; and P5 further emphasizes BC. The selected policies are simple enough to interpret, while covering the main cost–consolidation–inventory trade-offs identified during validation.

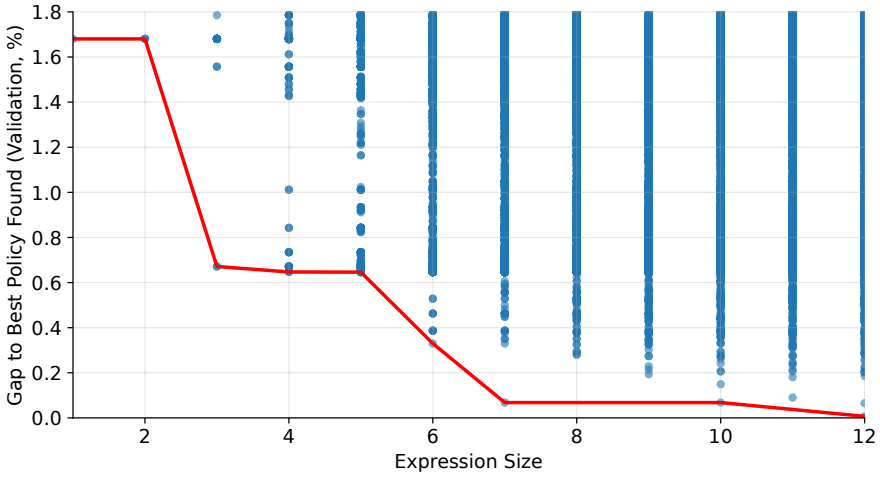


Figure 4.4: Validation performance versus expression size, highlighting the non-dominated front.

4.5.5.3 Candidate Policies Evaluation on Nominal Case

Table 4.2 reports the final out-of-sample evaluation on the nominal testing instance. The table compares the selected policies with the benchmark rule currently used in practice, using relative improvement in total fulfillment cost. Across all policies, the unfulfillment rate remains stable at 0.31%, so the reported cost differences are not obtained by sacrificing service level.

The results show that SC alone is insufficient, as P1 performs 2.36% worse than the benchmark. In contrast, policies that combine SC and BC perform much closer to the benchmark, and policies that additionally include SD outperform it. P5 achieves the best nominal testing result, with a 0.44% improvement over the incumbent rule. Over the six-month testing scale of the filtered dataset, this corresponds to approximately $76,000 in estimated fulfillment-cost savings.

The relative ranking of P1–P5 is preserved from validation to testing, suggesting that validation performance is informative for final out-of-sample selection. The results also show that improvement is achieved within the same feature space used by the current operational rule. It is important to stress that the benchmark policy itself was not provided to the GP algorithm. GP was therefore able to discover better-performing combinations of the same operational features through simulation-based search starting from randomly generated policies.

Moreover, any of the selected policies may be manually fine-tuned to better reflect managerial preferences or to identify higher-performing policies (e.g., by tuning policy coefficients or combining desirable properties of distinct policies). The tuned policies can also be re-tested to verify whether performance remains competitive. To illustrate this, Table 4.2 also includes P6, a manually refined variant inspired by the best-performing expressions. P6 smooths the SD term in the P5 structure, following the form suggested by P4. Although P6 does not outperform P5 in the nominal testing instance, it is retained for the perturbation analysis because it helps assess how small changes in the scoring rule

Table 4.2: Policies evaluated on the testing instance and their improvement over the benchmark.

Policy	Expression	Size	Improvement over Benchmark
Policies selected from the validation frontier			
P1	SC	1	−2.36%
P2	SC × BC	3	−0.05%
P3	SC × BC − log(SD)	6	0.16%
P4	SC × BC − log($\sqrt{\text{SD}}$)	7	0.40%
P5	SC × BC^2 + SC × BC − log(SD)	12	0.44%
Manual refinement			
P6	SC × BC^2 + SC × BC − log($\sqrt{\text{SD}}$)	13	0.32%
Benchmark: $10000 \times BC + SC - \frac{SD}{10000}$.			

affect robustness under different operating conditions.

4.5.5.4 Candidate Policies Evaluation under Inventory and Cost Perturbations

We finally evaluate the candidate policies under inventory and cost perturbations. The goal is to assess whether policies that perform well under nominal testing conditions remain attractive when operating conditions change. In fashion retail, inventory availability can vary substantially over the product life cycle: stock tends to be more abundant near the beginning of a season or after replenishment, and more constrained near the end of a season, during sales periods, or when popular items approach depletion. Fulfillment costs may also vary over time due to carrier price changes, peak-period surcharges, changes in service-level requirements, route availability, or shifts in the geographic composition of demand. We therefore use multiplicative perturbations from −50% to +50%, in steps of 10%, applied uniformly to: (i) inventory levels and exogenous inventory movements, and (ii) fulfillment costs.

Figure 4.5 reports three complementary views: the top-left panel shows fulfillment-cost reduction over the benchmark under inventory perturbations; the top-right panel shows the corresponding increase in unfulfilled demand; and the bottom-left panel shows cost reduction under cost perturbations. Negative improvements are omitted from the cost panels for readability. Unfulfillment under cost perturbations is not shown because differences remain below 0.01% across all policies and scenarios.

Under inventory perturbations, the policies with the largest cost reductions also tend to increase unfulfilled demand under scarcity. In the 0.5× inventory scenario, P3 and P5 reduce fulfillment cost by approximately 0.8% and 0.7%, respectively, relative to the benchmark, but P3 increases unfulfillment by approximately 0.42%, while P4 and P5 increase it by about 0.25%. As inventory availability increases, both cost and service-level differences converge, and all policies become similar to the benchmark in terms of unfulfilled demand. These results confirm that cost improvements under inventory scarcity must be interpreted jointly with service-level outcomes.

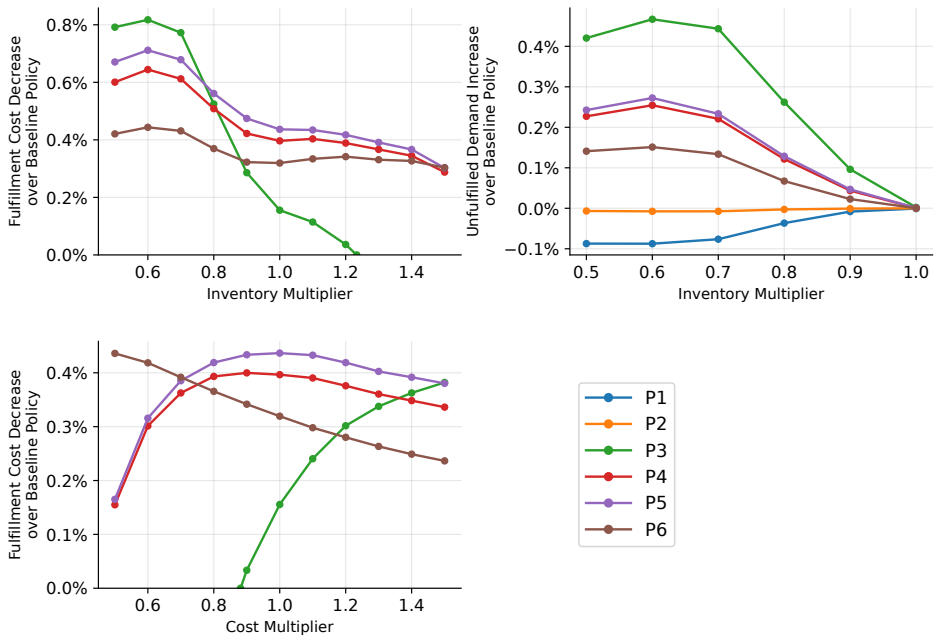


Figure 4.5: Robustness of the evaluated policies under cost and inventory perturbations on the testing instance.

The inventory perturbation results also clarify the role of SD. Policies that place greater emphasis on SD can improve cost efficiency under nominal or moderately constrained conditions, but they may also increase vulnerability to stockouts when inventory becomes scarce. By contrast, simpler policies that ignore SD, such as P1 and P2, tend to be slightly more stable in service terms under severe shortages, although they remain inferior in cost. This illustrates why the framework should be used not only to identify the lowest-cost policy, but also to understand the operating regimes in which each policy is acceptable.

Cost perturbations produce a different pattern. Because the perturbation changes the relative importance of SC and SD, different scoring structures become preferable under different cost regimes. Under low-cost scenarios, P6 achieves the largest improvements, around 0.40%–0.45%. Under nominal and moderately higher costs, P5 remains strongest, while at higher cost levels P3 becomes competitive, with improvements close to 0.4%. Since unfulfillment remains effectively unchanged under cost perturbations, these differences primarily reflect how each expression balances shipping cost and inventory depth.

Overall, no single policy dominates across all scenarios. P5, followed closely by P4, provides the strongest overall compromise under nominal and similar conditions, while other policies may be preferable in specific regimes. This reinforces the decision-support value of the framework: simulation does not only identify improved policies, but also reveals when those policies remain robust, when they create service-level trade-offs, and how policy choice should depend on operating conditions.

4.5.6 Practical and Managerial Implications

The case study shows how the proposed simulation-based optimization framework can support policy design, evaluation, and deployment in large-scale online fulfillment systems, from which some managerial implications have emerged.

Interpretable policies can be systematically improved. The results show that performance improvements can be achieved within a restricted and interpretable policy class. All evaluated policies are scoring rules composed of operationally meaningful features already used in practice. Nevertheless, GP identifies expressions that outperform the incumbent benchmark while remaining simple enough to inspect, communicate, and implement. This suggests that meaningful improvement may come not from replacing current decision logic with opaque models, but from systematically exploring alternative trade-offs within a deployable rule-based policy class.

Separation of feasibility and preference simplifies deployment. The framework enforces hard constraints inside the simulator and restricts optimized policies to ranking feasible options. This separation is important managerially: operational feasibility, eligibility rules, and non-negotiable business constraints remain controlled by the decision system, while GP searches only over preference rules among admissible actions. As a result, changes in business rules, node eligibility, or inventory constraints can be incorporated in the simulator without changing the policy representation or optimization workflow.

Offline optimization supports real-time execution. The computationally intensive steps (simulation, policy optimization, validation, and robustness testing) are performed offline. The resulting policies are lightweight scoring functions that can be evaluated in real time. This makes the approach compatible with high-throughput fulfillment environments, where allocation decisions must be made immediately but policy updates can be designed and tested periodically using recent data.

Scenario analysis exposes deployment trade-offs. The perturbation experiments show that nominal performance is not sufficient for policy selection. Policies that perform similarly under nominal conditions may behave differently under inventory scarcity or changing fulfillment-cost conditions. In particular, the case study shows that cost improvements under inventory scarcity may be accompanied by increased unfulfillment, highlighting a trade-off between efficiency and service level. Policies that place greater emphasis on SD can achieve lower fulfillment cost by exploiting inventory-depth signals, but may also become more vulnerable to shortages when inventory is scarce. Simpler policies that prioritize immediate cost and consolidation can provide more stable service outcomes under stress, even if they are less cost-effective in nominal conditions. Scenario-based evaluation therefore helps managers move beyond point estimates of performance and select policies according to robustness, service-level risk, and current operational priorities.

4.6. Conclusion

This paper proposes a simulation-based DSF for sequential fulfillment decisions in large-scale online retail systems characterized by dynamic inventory, complex feasibility rules,

and operational uncertainty. The framework focuses on evaluating and improving interpretable scoring-based allocation policies through simulation-based optimization.

At the modeling level, the framework represents order allocation as a sequential decision process with unit-level decisions, dynamic inventory evolution, feasible fulfillment options, and explicit unfulfilled-demand alternatives. At the methodological level, it combines a discrete-event simulator with GP-based policy optimization. This design separates feasibility handling from preference learning: the simulator enforces operational constraints, while GP searches over transparent scoring rules that rank feasible allocation options. As a result, the framework supports policy evaluation, offline policy improvement, validation, testing, and scenario-based robustness analysis within a deployable decision paradigm.

The case study, based on a large online fashion marketplace, demonstrates that the framework can be applied at a realistic scale using operational data. Despite a deliberately restricted policy representation, the framework identifies compact policies that outperform a baseline rule derived from current practice. The best-performing candidate improves nominal testing performance by 0.44%, corresponding to more than $150,000 on an annual basis. These improvements are achieved using the same operational features as the incumbent rule, reinforcing the practical relevance of improving a policy class compatible with current practice.

Beyond nominal performance, the results illustrate the value of simulation as a decision-support tool. Validation shows that much of the attainable improvement can be captured by relatively simple expressions, supporting the selection of policies that balance performance and interpretability. Scenario-based analyses further show that policy performance depends on operating conditions. In particular, inventory scarcity can create trade-offs between fulfillment-cost reductions and unfulfilled demand, while cost perturbations can change the relative attractiveness of different scoring structures. Making these trade-offs explicit is central to practical policy selection and would be difficult to achieve through single-scenario evaluation alone.

Several simplifying assumptions were adopted in the empirical implementation, including deterministic replay of historical demand and inventory movements, fixed exogenous inventory responses, and a single fulfillment-cost proxy. These choices were made to isolate the impact of allocation decisions and ensure tractability at scale. Future work could extend the simulator to incorporate stochastic scenario generation, endogenous inventory responses, additional business constraints, or richer cost and service-level objectives. Alternative policy-optimization methods could also be explored with the same simulation interface.

Appendix 4.A Detailed Problem Formulation

We model the problem as a sequential decision process using the notation of Powell (2019). The remainder of this section formally defines the core components of the model (*Decision Variables*, *State Variables*, *Exogenous Information*, *Transition Function*, and *Objective Function*), followed by the induced optimization problem—which together characterize

the order-by-order OF decision problem under dynamic inventory conditions and uncertainty. We then summarize the notation used in the model.

4.A.1 Decision Variables

Let t index the sequence of orders, where each order arrival defines a decision epoch, and let τ denote the (random) number of orders in the planning horizon. Order t may contain demand for one or multiple items $i \in I$, with no restriction on the number of units requested for each item. We represent the composition of order t by the set of unit–item pairs

$$U_t := \{(u, i_{u,t}) : u \text{ is a unit of order } t\}.$$

Units within an order are indexed in an arbitrary but fixed order, $u = 1, 2, \ldots, |U_t|$.

To specify a fulfillment plan, each unit must be assigned to a fulfillment node. Assignments may be made to any physical node $j \in J$ or to a fictitious node j', which represents unfulfilled demand, with $J' := J \cup \{j'\}$.

If all units of an order were allocated simultaneously, the number of possible allocations would be $|J'|^{|U_t|}$, which grows exponentially with the order size. To maintain a fixed and tractable action space, we instead decompose the fulfillment of order t into a sequence of $|U_t|$ unit-level decisions. At each decision step, the decision-maker selects exactly one node from the $|J'|$ available options.

We define the binary decision variable $x_{j,u,t}$ to equal 1 if fulfillment node $j \in J'$ is assigned to fulfill unit u of order t, and 0 otherwise. Each unit must be assigned to exactly one node:

$$\sum_{j \in J'} x_{j,u,t} = 1, \qquad \forall (u, i) \in U_t.$$

Let $s_{i,j,u,t}$ denote the available inventory of item i at node $j \in J$ immediately prior to the allocation of unit u of order t. Let κ_t represent an exogenous order attribute (e.g., geographic location and/or customer characteristics). Finally, let $g(j, \kappa_t) \in \{0, 1\}$ indicate whether fulfillment node $j \in J$ is eligible to serve orders with attribute κ_t. A decision is feasible only if the chosen node is both inventory-feasible and eligible for the order:

$$x_{j,u,t} = 1,\ j \in J,\ (u, i) \in U_t \implies s_{i,j,u,t} > 0 \text{ and } g(j, \kappa_t) = 1.$$

Equivalently, the feasible node set for unit u of order t is

$$J^{\text{feas}}_{t,u} = \left\{ j \in J : s_{i_{u,t},j,u,t} > 0,\ g(j, \kappa_t) = 1 \right\} \cup \{j'\}. \tag{4.A.1}$$

At decision step (t, u), the action consists of selecting a node in $J^{\text{feas}}_{t,u}$.

For the fictitious node j', we allow assignment regardless of inventory and eligibility.

4.A.2 State Variables

Following the granularity of decisions, we track the evolution of the system at the level of individual units within orders. We begin by describing the information available to the decision-maker immediately prior to any decision.

Inventory state Immediately before allocating unit u of order t, the available inventory for all items i and nodes $j \in J$ is represented by

$$\boldsymbol{s}_{u,t} = \big((s_{i,j,u,t})_i\big)_{j\in J}.$$

Order information Order t is characterized by the set of unit–item pairs U_t, together with the exogenous order attribute κ_t. We define

$$O_t := (U_t, \kappa_t).$$

Decision history We record past allocation decisions of the current order for reward computation purposes.

Decisions for earlier units of the current order t are

$$\mathcal{X}^{(t)}_{\text{units}<u} := \{(\tilde{u}, j_{\tilde{u},t}) : \ \tilde{u} < u\},$$

where $j_{\tilde{u},t}$ is the selected node for unit $\tilde{u}$, such that $x_{j_{\tilde{u},t},\tilde{u},t} = 1$.

Belief state We denote by $\mathcal{P}(\mathcal{Z})$ the set of probability distributions over a measurable space $\mathcal{Z}$. Let:

- $\mathcal{D}$ denote the space of possible demand realizations,
- $\mathcal{W}$ denote the space of possible exogenous inventory movements,
- $\mathcal{R}$ denote the space of possible reward (or cost) realizations.

At the beginning of order epoch t, the decision-maker holds a belief vector

$$b_t = (b_t^{\text{dem}}, b_t^{\text{inv}}, b_t^{\text{rew}}),$$

summarizing subjective probability distributions over future uncertainty:

1. Future demand:

$$b_t^{\text{dem}} \in \mathcal{P}(\mathcal{D}).$$

2. Exogenous inventory movements:

$$b_t^{\text{inv}} \in \mathcal{P}(\mathcal{W}).$$

3. Reward outcomes:

$$b_t^{\text{rew}} \in \mathcal{P}(\mathcal{R}).$$

We now assemble these information components into the system states. Because decisions are made sequentially at two levels (first across orders and then across the units within each order), we distinguish between the state immediately upon the arrival of an order and the state immediately after allocating a particular unit of that order.

Post–order-arrival state Immediately after the arrival of order t, before allocating any unit, the state is

$$S_t = (\boldsymbol{s}_{1,t},\, O_t,\, b_t).$$

For notational convenience, we define $S_{t,0} := S_t$.

Post–unit-allocation state After allocating unit u of order t, and, if the horizon has not ended, prior to the arrival of order $t+1$, the state is

$$S_{t,u} = (\boldsymbol{s}_{u+1,t},\, X^{(t)}_{\text{units}<u+1},\, O_t,\, b_t).$$

4.A.3 Exogenous Information

At the beginning of each order epoch t, the decision-maker observes exogenous information generated by two underlying stochastic processes: a demand process that governs the sequence of order arrivals, and an inventory-movement process that governs exogenous changes in inventory.

New order At order epoch t, the demand process reveals the new order characterized by O_t. Uncertainty in future order arrivals is represented by the belief component $b_t^{\text{dem}} \in \mathcal{P}(\mathcal{D})$.

Exogenous inventory movements Between the completion of decisions for order $t-1$ and the arrival of order t, the system may experience exogenous inventory movements. These movements occur irregularly and typically affect only a subset of item–node pairs.

To model these events, we define the inventory-change space

$$\mathcal{W} = \{(i, j, \delta) : i \in I,\ j \in J,\ \delta \neq 0\},$$

where each triple (i, j, δ) represents a nonzero inventory change δ of item i at node j (positive for increases such as replenishments or inbound transshipments, negative for decreases such as external sales, shrinkage, or spoilage).

The realized exogenous inventory movements observed at the beginning of order epoch t (i.e., between $t-1$ and t) are represented by the finite set

$$\Delta_t \subseteq \mathcal{W},$$

whose elements identify exactly which inventory changes occurred and by how much. Uncertainty over future inventory changes is encoded in the belief component $b_t^{\text{inv}} \in \mathcal{P}(\mathcal{W})$.

4.A.4 Transition Function

The evolution of the system is driven by the sequence of unit-level allocation decisions $x_{j,u,t}$ and by the exogenous information given by order arrivals O_t and inventory-change events Δ_t. Consequently, transitions occur both within an order and between consecutive orders.

Within-order transitions The state immediately after the arrival of order t is S_t. The allocation of its units proceeds sequentially:

1. **Transition from S_t to $S_{t,1}$.** After observing S_t and allocating the first unit $(u = 1, i_{1,t})$, the system transitions to $S_{t,1}$. Inventory and decision history update as described below, with the allocation of unit $u = 1$ reflected in $s_{2,t}$ and $\mathcal{X}^{(t)}_{\text{units}<2}$.
2. **Transition from $S_{t,u-1}$ to $S_{t,u}$.** Suppose that the system is in state $S_{t,u-1}$, with $2 \leq u < |U_t|$. After allocating unit u of order t to node j, the system transitions to $S_{t,u}$.

Two components evolve during within-order transitions:

- **Inventory update.** If $(u, i^*) \in U_t$ is allocated to a physical node $j^* \in J$, one unit of item i^* is consumed:

$$s_{i,j,u+1,t} = \begin{cases} s_{i^*,j^*,u,t} - 1, & \text{if } (i,j) = (i^*, j^*), \\ s_{i,j,u,t}, & \text{otherwise.} \end{cases} \tag{4.A.2}$$

  No inventory is consumed if the fictitious node j' is selected.

- **Decision history update.** The newly selected action is appended to the current-order decision history:

$$\mathcal{X}^{(t)}_{\text{units}<u+1} = \mathcal{X}^{(t)}_{\text{units}<u} \cup \{(u, j) : x_{j,u,t} = 1\}. \tag{4.A.3}$$

All other state components remain unchanged during within-order transitions. This process continues until the final unit $u = |U_t|$ is allocated, yielding the state $S_{t,|U_t|}$. This history is discarded at the end of the order and does not carry over to state S_{t+1}.

Between-order transitions Once all units of order t have been allocated and the system is in state $S_{t,|U_t|}$, it transitions to the next post–order-arrival state S_{t+1}.

Two components govern this transition:

- **Exogenous inventory changes.** Before the arrival of order $t+1$, the inventory adjusts according to the realized inventory-change events Δ_{t+1}:

$$s_{i,j,1,t+1} = s_{i,j,|U_t|+1,t} + \sum_{(i',j',\delta) \in \Delta_{t+1}} \mathbf{1}\{(i', j') = (i, j)\}\, \delta. \tag{4.A.4}$$

- **Arrival of the next order.** The demand process reveals O_{t+1}.

The belief state updates by the agent's learning rule, yielding b_{t+1}. The new state becomes

$$S_{t+1} = (\boldsymbol{s}_{1,t+1},\, O_{t+1},\, b_{t+1}).$$

If $t = \tau$, no further orders arrive, and the system terminates.

4.A.5 Objective Function

At each unit-allocation decision, the reward process $\mathcal{R}$ generates a reward composed of (i) deterministic, fully known components and (ii) uncertain components whose distributions are captured by the belief state b_t^{rew}.

Unit-level reward components. For unit u of order t, let $R_{t,u}^{\text{known}}$ denote the deterministic reward (or cost) associated with assigning this unit to a node $j \in J'$. This term may include both variable components (e.g., unit-level shipping costs) and fixed components (e.g., a fixed fulfillment charge incurred when shipping at least one unit from a given node to the customer).

Let $R_{t,u}^{\text{unc}}$ denote the random reward component arising from uncertain operational outcomes affecting this unit (e.g., delivery delay, realized service quality). Conditioned on the state and action at step (t,u), its distribution is represented by the belief component $b_t^{\text{rew}} \in \mathcal{P}(\mathcal{R})$.

Because uncertain outcomes are not observed at decision time and only their distribution matters for planning, we work with their expected values. We therefore define the *expected unit-level reward*

$$\bar{R}_{t,u} = R_{t,u}^{\text{known}} + \mathbb{E}_{R_{t,u}^{\text{unc}} \sim b_t^{\text{rew}}}\left[R_{t,u}^{\text{unc}}\right]. \tag{4.A.5}$$

Order-level and horizon-level rewards. The reward for order t is the sum of its expected unit-level rewards,

$$R_t = \sum_{u=1}^{|U_t|} \bar{R}_{t,u},$$

and the total reward accumulated over the (random) horizon τ is

$$\sum_{t=1}^{\tau} R_t.$$

This formulation is deliberately flexible: it does not impose a specific structure for either the deterministic or uncertain reward components and accommodates both unit-level and order-level dependencies arising from the allocation history.

4.A.6 Optimization Problem

A policy π is a (possibly stochastic) mapping from the current state to a feasible action. At each unit-allocation step (t, u), the selected node is given by

$$j_{t,u} = \pi(S_{t,u-1}) \in J_{t,u}^{\text{feas}}. \tag{4.A.6}$$

Given the system dynamics and reward structure described above, the objective is to choose a policy that maximizes expected cumulative reward over the horizon.

Formally, the decision-maker solves

$$\max_{\pi} \mathbb{E}^{\pi}\left[\sum_{t=1}^{\tau}\sum_{u=1}^{|U_t|}\bar{R}_{t,u}\right],$$

where (1) $\bar{R}_{t,u}$ is defined from b_t^{rew} as in the objective function, and (2) the outer expectation is taken with respect to the stochastic evolution of the system under policy π, including demand arrivals, exogenous inventory movements, belief updates, and the random length of the horizon τ.

4.A.7 Model Summary

Table 4.A.1 summarizes the indices, sets, variables, and exogenous inputs used in the model.

Symbol	Description
Indices and Horizon	
t	Order index (decision epoch)
u	Unit index within order t
τ	Random number of orders in the planning horizon
Sets	
I	Set of items
J	Set of physical fulfillment nodes
J'	Fulfillment nodes including fictitious node j' for unfulfilled demand, with $J' = J \cup \{j'\}$
$J_{t,u}^{\text{feas}}$	Feasible node set for unit u of order t
U_t	Set of unit–item pairs $(u, i_{u,t})$ comprising order t
Decision Variables	
$x_{j,u,t} \in \{0,1\}$	Equals 1 if unit u of order t is assigned to node $j \in J'$
State Variables	
$s_{i,j,u,t}$	Inventory of item i at node j before allocating unit u
$\boldsymbol{s}_{u,t}$	Inventory vector before allocating unit u of order t
κ_t	Exogenous order attribute, such as delivery location or customer characteristics
$g(j, \kappa_t)$	Eligibility indicator, equal to 1 if node j may serve orders with attribute κ_t
$O_t = (U_t, \kappa_t)$	Order information, including item requests and exogenous attributes
$\mathcal{X}^{(t)}_{\text{units}<u}$	Allocation history for units $1, \ldots, u-1$ of order t
b_t	Belief state regarding uncertainty, with $b_t = (b_t^{\text{dem}}, b_t^{\text{inv}}, b_t^{\text{rew}})$
Exogenous Inputs	
Δ_t	Realized exogenous inventory changes between orders $t-1$ and t
$\mathcal{P}(\mathcal{Z})$	Set of probability distributions over $\mathcal{Z}$
$\mathcal{W}$	Space of possible nonzero inventory-change events
$\mathcal{D}$	Space of possible demand realizations
$\mathcal{R}$	Space of possible reward realizations
Rewards	
$R_{t,u}^{\text{known}}$	Deterministic reward, or cost, for allocating unit u
$R_{t,u}^{\text{unc}}$	Random reward component for unit u
$\bar{R}_{t,u}$	Expected unit-level reward under belief b_t^{rew}
R_t	Total reward for order t

Table 4.A.1: Summary of model notation.

Bibliography

Jason Acimovic and Stephen C. Graves. Making better fulfillment decisions on the fly in an online retail environment. *Manufacturing & Service Operations Management*, 17(1): 34–51, 2015. doi: 10.1287/MSOM.2014.0505.

John M. Andrews, Vivek F. Farias, Aryan I. Khojandi, and Chad M. Yan. Primal–dual algorithms for order fulfillment at Urban Outfitters, Inc. *INFORMS Journal on Applied Analytics*, 49(5):355–370, 2019. doi: 10.1287/INTE.2019.1013.

Arash Asadpour, Xuan Wang, and Jiawei Zhang. Online resource allocation with limited flexibility. *Management Science*, 66(2):642–666, 2019. doi: 10.1287/MNSC.2018.3220.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021. doi: 10.1016/j.ejor.2020.07.063.

Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, 2020. doi: 10.1287/mnsc.2018.3253.

Dimitris Bertsimas, Jack Dunn, and Nishanth Mundru. Optimal prescriptive trees. *INFORMS Journal on Optimization*, 1(2):164–183, 2019. doi: 10.1287/ijoo.2018.0005.

Jürgen Branke, Su Nguyen, Christoph W. Pickardt, and Mengjie Zhang. Automated design of production scheduling heuristics: A review. *IEEE Transactions on Evolutionary Computation*, 20(1):110–124, 2016. doi: 10.1109/TEVC.2015.2429314.

Junhui Cai, Ran Chen, Martin J. Wainwright, and Linda Zhao. Doubly high-dimensional contextual bandits: An interpretable model for joint assortment-pricing, 2023. URL https://arxiv.org/abs/2309.08634. arXiv preprint.

F. Coelho, S. Relvas, and A. P. Barbosa-Póvoa. Simulation-based decision support tool for in-house logistics: The basis for a digital twin. *Computers & Industrial Engineering*, 153:107094, 2021. doi: 10.1016/j.cie.2020.107094.

Cristiane Ferreira, Gonçalo Figueira, and Pedro Amorim. Effective and interpretable dispatching rules for dynamic job shops via guided empirical learning. *Omega*, 111:102643, 2022. doi: 10.1016/j.omega.2022.102643.

Santiago Gallino, Antonio Moreno, and Guillaume Roels. How to fulfill online orders without breaking the bank: Optimal store fulfillment strategies for omnichannel retailers. *Manufacturing & Service Operations Management*, 23(4):855–872, 2021.

Bissan Ghaddar, Nizar Sakr, and Yaw Asiedu. Spare parts stocking analysis using genetic programming. *European Journal of Operational Research*, 252(1):136–144, 2016. doi: 10.1016/j.ejor.2015.12.041.

Pavithra Harsha, Ashish Jagmohan, Jayant Kalagnanam, Brian Quanz, and Divya Singhvi. Deep policy iteration with integer programming for inventory management, 2025. URL https://arxiv.org/abs/2112.02215. Preprint.

Stefanus Jasin and Amitabh Sinha. An LP-based correlated rounding scheme for multi-item ecommerce order fulfillment. *Operations Research*, 63(6):1336–1351, 2015. doi: 10.1287/OPRE.2015.1441.

Tho V. Le and Ruoling Fan. Digital twins for logistics and supply chain systems: Literature review, conceptual framework, research potential, and practical challenges. *Computers & Industrial Engineering*, 187:109768, 2024. doi: 10.1016/j.cie.2023.109768.

Rui L. Lopes, Gonçalo Figueira, Pedro Amorim, and Bernardo Almada-Lobo. Cooperative coevolution of expressions for (r,Q) inventory management policies using genetic programming. *International Journal of Production Research*, 58(2):509–525, 2020. doi: 10.1080/00207543.2019.1597293.

Sean Luke. ECJ evolutionary computation library, 1998. URL https://cs.gmu.edu/~eclab/projects/ecj/. Software library.

Artur M. Pereira, J. Antão B. Moura, Evandro De B. Costa, Thales Vieira, André R. D. B. Landim, Eirini Bazaki, and Vanissa Wanick. Customer models for artificial intelligence-based decision support in fashion online retail supply chains. *Decision Support Systems*, 158:113795, 2022. doi: 10.1016/j.dss.2022.113795.

Warren B. Powell. A unified framework for stochastic optimization. *European Journal of Operational Research*, 275(3):795–821, 2019. doi: 10.1016/j.ejor.2018.07.014.

Sushil Punia, Surya Prakash Singh, and Jitendra K. Madaan. From predictive to prescriptive analytics: A data-driven multi-item newsvendor model. *Decision Support Systems*, 136:113340, 2020. doi: 10.1016/j.dss.2020.113340.

Sérgio Vasconcelos, Gonçalo Figueira, and Bernardo Almada-Lobo. Order allocation in online retail: Classification and literature review. *European Journal of Operational Research*, 333(1):1–18, 2026. doi: 10.1016/j.ejor.2025.11.018.

Yihua Wang and Stefan Minner. Data-driven multi-location inventory placement in digital commerce. *Computers & Industrial Engineering*, 200:110842, 2025. doi: 10.1016/j.cie.2024.110842.

Yongbo Xiao, Frank Y. Chen, and Jian Chen. Optimal inventory and dynamic admission policies for a retailer of seasonal products with affiliate programs and drop-shipping. *Naval Research Logistics*, 56(4):300–317, 2009. doi: 10.1002/NAV.20346.

Chapter 5

General discussion, conclusions, and future research

This thesis studied OAOR, with particular emphasis on online marketplace settings. In online retail, fulfillment shifts from a customer-selected process, typical of traditional retail, to a retailer- or platform-directed decision in which incoming demand must be assigned to available inventory across a fulfillment network. In marketplace settings, this decision becomes especially relevant as fulfillment sources often correspond to third-party vendors, and allocation decisions may affect not only cost and customer satisfaction, but also vendor satisfaction.

The thesis addressed this problem through three article-based chapters, each positioned at a different level of analysis. Chapter 2 provided the conceptual foundation by defining OAOR, formulating a baseline problem, classifying problem extensions, and organizing solution methods. Chapter 3 then developed a marketplace-specific modeling extension by introducing the MIOFP-OM, in which vendor satisfaction is represented through allocation targets and shortfall penalties. Chapter 4 moved from modeling to operational support by proposing a modular, data-driven, simulation-based framework for evaluating and improving deployable allocation policies using operational data.

Together, the three studies show that order allocation should be understood as a sequential long-term cost-minimization decision, a mechanism for managing marketplace stakeholder outcomes, and a practical policy deployment problem. The literature review shows the breadth and fragmentation of the OAOR field; the marketplace model shows that introducing vendor-related objectives creates explicit fulfillment-cost/vendor-satisfaction trade-offs; and the DSF shows how allocation policies can be tested, optimized, and interpreted before deployment in realistic operational environments.

This concluding chapter builds on these findings. Section 5.1 answers the research questions stated in Chapter 1. Section 5.2 discusses the connections between the three studies. Sections 5.3 and 5.4 then discuss managerial implications and future research directions.

5.1. Answers to the research questions

Chapter 1 formulated one overarching research question (ORQ) and three specific research questions. This section answers each question using the evidence extracted from the three article-based chapters. The answers are presented separately, before being brought together in response to the overarching research question.

RQ1 asked: How can OAOR be systematically characterized, and what problem variants, solution approaches, and research gaps emerge from the literature?

Chapter 2 answers this question by defining OAOR as a distinct operational decision and by separating it from broader e-fulfillment decisions such as network design, warehouse design, delivery service design, pricing, forecasting, transportation planning, and inventory-capacity planning. This distinction is important because order allocation is often studied under different names and sometimes jointly with other operational decisions, making it difficult to identify the core decision being optimized.

The systematic characterization developed in Chapter 2 shows that OAOR can be understood through two complementary classification lenses. The first lens concerns the problem setting: allocation timing, order composition, fulfillment-network structure, inventory dynamics, objective functions, and the interaction between order allocation and adjacent decisions. The second lens concerns the solution approach: optimization mechanism, policy class, use of future information, and incorporation of learning. This classification makes it possible to compare studies that would otherwise appear disconnected and to identify gaps both in problem settings and in solution methods. In particular, the review identifies under-addressed problem variants, including order-specific allocation delay, product substitution, delivery delays and errors, order returns, and external inventory movements, while also highlighting opportunities in multi-objective optimization, reinforcement-learning-based methods, and broader benchmarking of allocation policies.

RQ2 asked: How does the order allocation problem change in online marketplaces when vendor satisfaction is explicitly considered as a fulfillment objective?

Chapter 3 answers this question by showing that, in marketplace settings, fulfillment nodes should not always be treated as neutral facilities controlled by a single retailer. When fulfillment sources are vendors, allocation decisions can affect the distribution of sales opportunities across marketplace partners. This changes the structure of the order allocation problem: the platform must still manage fulfillment costs and unfulfilled demand, but it may also need to account for whether vendors receive allocations that are consistent with commercial expectations or relationship-management goals.

The MIOFP-OM formalizes this change by extending the classical MIOFP to a setting with vendor-owned inventory, finite selling seasons, dynamic order arrivals, and vendor allocation targets. Vendor satisfaction is represented through end-of-season shortfall penalties: when a vendor receives fewer allocations than its target, the model assigns a penalty that reflects the dissatisfaction associated with that shortfall. This formulation makes the marketplace-specific trade-off explicit. Allocations that minimize immediate fulfillment costs may systematically favor some vendors and leave others below their allocation targets, while satisfaction-aware allocations can reduce vendor shortfalls but may increase fulfillment costs or affect demand fulfillment.

The computational results show that the fulfillment-cost/vendor-satisfaction trade-off depends on (1) the pressure created by the targets and (2) the operational structure of the marketplace. Increasing target ambition raises remaining shortfall, but sharply reduces the proportion that allocation policies can eliminate. Increasing the shortfall penalty initially

strengthens target steering, but its marginal effect eventually saturates. Target granularity also matters. Vendor-wide targets are easier to attain because allocations across items contribute to a common requirement, whereas item–vendor targets provide stronger control over assortment-level outcomes, and produce significant trade-offs when specific items have targets that are more difficult to meet. The policy comparison shows that myopic target awareness alone is not Pareto-efficient. DCL generally provides the more fulfillment-cost-efficient shortfall reductions, while CF-PD often achieves larger reductions at higher cost. R-D is less competitive in the cost-aligned reference setting but becomes more relevant when within-order coordination and restricted item–vendor substitution matter. CF-LP is often dominated and additionally requires repeated online optimization. Overall, non-myopic policies are most valuable when the target shortfall is both significant and controllable, i.e., when current allocations meaningfully affect future fulfillment opportunities.

RQ3 asked: How can order allocation policies be evaluated and improved using operational data while preserving interpretability, robustness, and deployment feasibility?

Chapter 4 answers this question by proposing a modular, data-driven, simulation-based DSF. The framework transforms historical operational data and business rules into simulation instances, evaluates candidate allocation policies under controlled scenarios, and uses the same simulator as an evaluation engine for policy optimization. This creates an offline analytics layer through which policies can be tested and improved before being considered for deployment in the live fulfillment system.

The framework can accommodate alternative demand, inventory, cost, and operational scenarios; in the case study, robustness is demonstrated explicitly through perturbations to inventory availability and fulfillment costs. This is important because many real allocation policies are manually designed, rule-based, and implemented directly in production, with their impact assessed only after deployment. By contrast, the proposed framework enables systematic ex ante evaluation, stress testing, and robustness analysis. It therefore reduces the risk of changing allocation logic and provides a structured way to understand the trade-offs induced by different policies.

The framework's policy improvement layer focuses on interpretable score-based policies and on using GP to optimize them based on simulation-based performance feedback. This choice is important for deployment feasibility: rather than replacing existing decision logic with an opaque black-box model, the framework searches within a policy class that resembles the scoring rules often used in practice. In the fashion marketplace case study, this approach demonstrates that operational data can be used not only for reporting or forecasting, but also to generate compact, interpretable, and improved allocation policies. In the nominal testing instance, the best compact policy reduces fulfillment cost by 0.44%, corresponding to more than \$150,000 annually, without degrading the nominal unfulfillment rate. More broadly, the framework is configurable: different objectives, constraints, policy representations, and scenario-generation procedures can be easily incorporated depending on the operational context.

ORQ asked: How can OAOR, particularly in online marketplace settings, be structured, modeled, evaluated, and supported under realistic operational conditions?

The three article-based chapters jointly answer this question by showing that order allocation requires four complementary layers of analysis. It can be *structured* by defining OAOR as a distinct decision problem and classifying its problem settings and solution methods; it can be *modeled* in marketplace settings by representing vendor-related objectives through allocation targets and shortfall penalties; it can be *evaluated* through systematic comparison of allocation policies under controlled computational experiments; and it can be *supported* through simulation-based decision-support tools that use operational data to evaluate and improve deployable policies. Table 5.1 summarizes these answers, makes explicit the related thesis contributions to literature and practice, and the remaining issues that motivate the synthesis and future research agenda developed in the rest of this chapter.

5.2. Cross-chapter synthesis

This section discusses what emerges when the three article-based chapters are read together. The synthesis developed identifies the common themes that connect the three studies: problem structuring, marketplace stakeholder trade-offs, policy representation, and the relationship between controlled modeling and operational decision support.

The first cross-chapter insight is that problem structuring is a prerequisite for meaningful model and policy design. Chapter 2 shows that OAOR has been studied under different names, assumptions, extensions, and solution mechanisms. By defining a baseline problem and classifying extensions and solution methods, it provides the language needed to distinguish what changes across order allocation contexts. Chapters 3 and 4 then use this structure in different ways. Chapter 3 emphasizes a marketplace-specific change in the objective structure of the MIOFP, by adding vendor allocation targets and shortfall penalties. Chapter 4 emphasizes the operationalization of order allocation policies by focusing on deployability, operational data, scenario construction, simulation-based evaluation, and interpretable policy representations. In this sense, Chapter 2 does not provide a single model that is directly extended in the following chapters. Rather, it provides a reference structure for identifying which elements of the order allocation problem are being emphasized, modified, or operationalized in each study.

The second insight is that marketplace order allocation is not just "order allocation with a different type of fulfillment node". In vertically integrated retail networks, fulfillment nodes are typically modeled as facilities controlled by the retailer. In marketplace settings, however, fulfillment nodes may correspond to vendors that manage their own inventory, voluntarily participate in the platform, and care about the allocations they receive. Chapter 3 makes this distinction explicit by modeling vendor satisfaction through allocation targets and shortfall penalties. Chapter 4 complements this perspective by showing that real marketplace fulfillment also imposes practical requirements: allocation policies must be interpretable, fast, robust to imperfect operational information, and compatible with ex-

Table 5.1: Answers to the research questions, related thesis contributions, and remaining research issues.

Research question	Answer	Thesis contribution	Remaining issues
RQ1. How can OAOR be systematically characterized?	OAOR can be characterized as a distinct operational decision and organized through problem-setting dimensions and solution-method dimensions.	A unified definition and classification framework comprising a baseline OAOR problem, problem-extension dimensions, and solution-method dimensions, providing a common vocabulary for comparing a fragmented literature and identifying research gaps.	Underexplored problem settings and extensions, including external inventory movements, order returns, richer demand- and fulfillment-process-management decisions, and broader benchmarking of solution methods.
RQ2. How does OAOR change when vendor satisfaction is considered?	Marketplace order allocation becomes a multi-stakeholder allocation problem in which vendor-owned fulfillment nodes have allocation outcomes that matter in their own right.	The MIOFP-OM, which represents vendor satisfaction through allocation targets and shortfall penalties, together with a common experimental setting for quantifying fulfillment-cost/vendor-satisfaction trade-offs and comparing multiple allocation-policy classes.	Richer vendor behavior, heterogeneous penalties, dynamic inventory commitment, and broader marketplace settings.
RQ3. How can policies be evaluated and improved using operational data?	Policies can be evaluated and improved offline through a modular simulation-based DSF using operational data, controlled scenarios, interpretable policy representations, and simulation-based optimization.	A configurable DSF connecting operational-data processing, simulation-instance construction, policy evaluation, robustness analysis, and GP-based optimization of interpretable score-based policies.	Live deployment, richer objectives, stochastic scenario generation, and integration with marketplace-specific formulations.

isting business logic. Together, the two chapters show that marketplace order allocation requires both richer objective modeling and careful attention to deployability.

The third insight concerns long-term trade-offs. Across the thesis, order allocation is repeatedly shown to be more than immediate fulfillment-cost minimization. In Chapter 2, this appears through the classification of problem extensions and objectives that connect order allocation with inventory management, demand management, and fulfillment process management. In Chapter 3, the trade-off is made explicit through the computational analysis of vendor prioritization: reducing vendor target shortfalls can require higher fulfillment costs, and the strength of this trade-off depends on target ambition, penalty intensity, target granularity, demand structure, inventory positioning, and network flexibility. In Chapter 4, the trade-off appears in operational form: allocation policies must balance performance improvement with interpretability, robustness, data requirements, and implementation feasibility. The common lesson is that order allocation policies should be evaluated by the objective value they produce in a stylized model, and, more importantly, by the organizational and operational conditions under which they can be used.

The fourth insight concerns policy representation and comparison. Chapter 2 shows that the OAOR literature contains several solution mechanisms and policy classes, but that comparison across policy families is often limited. Chapter 3 responds to this limitation by adapting and comparing multiple policy classes within the same marketplace formulation, including myopic, cost-function, randomized, and learning-based policies. The results discussed in Section 5.1 show that the best policy depends on the metric used to assess the trade-off: DCL generally provides more cost-efficient shortfall reduction, whereas CF-PD may achieve stronger steering at a higher fulfillment-cost premium. Chapter 4 then studies a different but complementary policy question: rather than comparing several literature policy classes, it focuses on one deployable class, score-based policies, and shows how such policies can be evaluated and improved using simulation and GP. Taken together, the thesis argues for matching policy representations to the structure of the fulfillment environment and to the practical requirements of the decision context.

The fifth insight is that controlled modeling and operational decision support serve complementary purposes. Chapter 3 uses generated instances and explicit modeling assumptions to isolate the effect of vendor satisfaction on allocation decisions. This controlled setting is useful because it allows target ambition, penalty intensity, target granularity, and scenario characteristics to be varied systematically. Chapter 4, in contrast, starts from operational data and business requirements. Its value lies in showing how policies can be assessed under realistic data conditions, how simulation can reproduce relevant fulfillment dynamics, and how optimization can be used to improve interpretable rules before deployment. The two approaches are not alternative: the modeling contribution in Chapter 3 identifies objectives and mechanisms that matter in marketplace fulfillment; the decision-support contribution in Chapter 4 provides a framework through which such objectives could later be incorporated, tested, and operationalized.

This synthesis also clarifies the role of assumptions in the thesis. Chapter 3 deliberately simplifies some operational features, such as replenishment during the horizon, multi-node vendors, and dynamic vendor inventory commitment, in order to study the vendor-satisfaction mechanism clearly. Chapter 4 makes different simplifications, such as focusing

on score-based policies, deterministically replaying historical demand and inventory movements, treating exogenous inventory responses as fixed, and using a single fulfillment-cost proxy. These assumptions limit the scope of each chapter, but they also define a coherent research trajectory. The thesis first clarifies the problem space, then studies a marketplace-specific objective in a controlled model, and finally develops a practical framework for evaluating and improving allocation policies under realistic operational constraints.

Table 5.2 summarizes the main cross-chapter themes discussed in this section. Overall, the thesis contributes an integrated approach to order allocation in online retail. At the conceptual level, it defines and structures OAOR as a distinct operational decision. At the modeling level, it extends the problem to marketplace settings by explicitly representing vendor-related allocation outcomes. At the methodological and practical levels, it compares alternative policy representations and develops a simulation-based framework for evaluating and improving interpretable allocation policies using operational data. The contribution of the thesis, therefore, lies in connecting problem structuring, marketplace-specific modeling, policy comparison, and deployment-oriented decision support within a unified research trajectory.

5.3. Managerial implications

The findings of this thesis suggest that order allocation should be treated as an operational decision with tactical and strategic impacts. Although each decision determines which node fulfills the current demand, repeated allocations shape fulfillment costs, inventory availability, customer service, and, in marketplace settings, vendor-related outcomes. Allocation logic should therefore be designed and evaluated based on these cumulative effects.

For marketplace operators, one important implication is that vendor-related objectives should be made explicit. If a platform cares about vendor satisfaction or compliance with commercial agreements, these concerns should be translated into measurable decision criteria, such as allocation targets, shortfall penalties, or service-level requirements. The vendor-satisfaction study shows the risk of leaving these objectives implicit: allocation policies focused only on immediate fulfillment efficiency may systematically favor low-cost or well-positioned vendors, while other vendors receive fewer sales opportunities than expected. Conversely, rigid priority rules may overcorrect in the opposite direction, satisfying vendor-prioritization logic while incurring unnecessary fulfillment costs and reducing future allocation flexibility. Modeling vendor satisfaction through targets and penalties, therefore, gives managers a way to calibrate how strongly vendor outcomes should influence allocation decisions and to quantify the cost of doing so.

A second implication is that there is no universally best allocation policy. DCL generally provides the more cost-efficient reduction of vendor-target shortfall, while CF-PD may achieve larger reductions at a higher fulfillment-cost premium. Correlated rounding becomes more relevant when within-order coordination and restricted substitution are operationally important, whereas repeated live optimization makes CF-LP less attractive

Table 5.2: Cross-chapter synthesis themes.

Synthesis theme	Chapter 2	Chapter 3	Chapter 4
Problem structuring	Defines OAOR, baseline problem, extensions, and solution-method dimensions.	Uses the OAOR structure to extend MIOFP to marketplace vendor objectives.	Uses the OAOR structure to define a configurable simulation-based policy-evaluation setting.
Marketplace specificity	Identifies underexplored marketplace-related issues, including vendor-demand allocation and external inventory movements.	Models vendor-owned inventory and vendor satisfaction explicitly.	Instantiates decision support in a real fashion marketplace with heterogeneous vendors and operational constraints.
Trade-offs	Shows that OAOR objectives extend beyond immediate fulfillment cost.	Quantifies fulfillment-cost/vendor-satisfaction trade-offs under targets and penalties.	Studies performance, robustness, interpretability, and deployability trade-offs in policy evaluation and optimization.
Policy representation	Classifies solution methods by mechanism, policy class, future information, and learning.	Adapts and compares myopic, cost-function, randomized, and learning-based policies.	Focuses on interpretable score-based policies optimized through simulation-based GP.
Path to deployment	Calls for broader policy benchmarking to clarify which approaches are most suitable across problem contexts.	Highlights marketplace objectives that future deployable systems may need to represent.	Provides an offline analytics framework for testing and improving policies before deployment.

for time-critical execution. However, the practical attractiveness of a policy depends on its simulated performance, as well as on information, implementation, and governance requirements. Myopic policies are simple and do not require explicit demand forecasts; CF-LP and rounding policies may require demand or order-composition forecasts; CF-PD and DCL require calibration or simulation-based training; and policies with high online computational burden may be less attractive even when they improve over simple benchmarks. Chapter 4 adds an important practical perspective to this discussion: score-based policies can provide a useful compromise between performance and deployability, because they often resemble existing business rules, remain interpretable, and can still be improved through simulation-based optimization. Managers should therefore select allocation policies by jointly considering performance, data availability, computational requirements, interpretability, robustness, and compatibility with existing decision logic.

Finally, the DSF in Chapter 4 suggests that simulation can reduce the risk of changing allocation logic in production. Historical order, inventory, cost, and fulfillment data can be transformed into simulation instances to evaluate incumbent rules, test manually designed alternatives, and optimize interpretable score-based policies before deployment. The same environment can be used to stress-test policies under conditions such as inventory scarcity, unusually abundant inventory, demand peaks, or cost perturbations. Managers can therefore assess whether performance improvements are robust across operating conditions and identify policies that create unacceptable outcomes in adverse scenarios. Allocation-policy changes can consequently be supported by an offline evaluation environment that combines operational data, business constraints, simulation, and interpretable policy improvement.

5.4. Future research agenda

The findings of this thesis open several directions for future research. This section discusses how the scope, assumptions, and empirical settings of the three article-based chapters point toward a broader research agenda.

A first direction concerns the continued development of OAOR as a research field. Future work can build on the classification framework in Chapter 2 by studying underexplored problem extensions, such as external inventory movements, order returns, demand-management interactions, fulfillment-process decisions, and richer combinations of order allocation with adjacent operational decisions. For example, online grocery fulfillment could motivate perishability-aware order allocation, in which fulfillment decisions account not only for cost and availability but also for remaining shelf life, freshness, cold-chain constraints, and the risk of quality deterioration before delivery (Amorim et al., 2013; Hübner et al., 2016). Similarly, return-aware order allocation could study how fulfillment choices affect the probability, condition, and cost of returned products, for instance, when certain routes, packaging choices, or fulfillment nodes are associated with higher damage or reverse-logistics costs. This would connect OAOR with the broader product-returns and reverse-logistics literature while introducing new fulfillment-specific decision criteria.

Another important direction related to the development of OAOR is the construction of comparable benchmark settings. Such benchmarks would make it easier to compare policy classes under common assumptions, rather than evaluating each new method only against a simple myopic baseline or within a highly specific problem variant.

A second direction concerns richer marketplace modeling. Chapter 3 introduced vendor satisfaction through allocation targets and shortfall penalties. Future work could consider heterogeneous penalties across vendors or items, nonlinear dissatisfaction functions, and learning-based calibration of targets from historical vendor behavior or contractual arrangements. Another promising direction is to treat vendor satisfaction as a separate objective during policy optimization, rather than incorporating it into the scalar objective used to fit or train each policy. In the present study, each policy is optimized using a single objective that combines fulfillment costs and vendor-target shortfalls via fixed penalty parameters; the resulting policies are then evaluated in terms of their fulfillment-cost and vendor-satisfaction trade-offs, including via Pareto analysis. A complete multi-objective approach would instead optimize these criteria jointly and directly construct a set of non-dominated policies spanning different trade-offs among fulfillment efficiency, demand fulfillment, and vendor satisfaction. This could be achieved through scalarization schemes, ε-constraint methods, or evolutionary multi-objective algorithms (Deb et al., 2002). Such an approach would provide decision-makers with a more explicit view of the attainable trade-off frontier, without requiring the relative importance of vendor outcomes to be fixed through penalty parameters before policy optimization.

A third direction is integrating marketplace-specific objectives into DSFs. Chapter 3 studies vendor satisfaction in a controlled experimental setting, while Chapter 4 develops a data-driven framework for evaluating and improving deployable allocation policies. A natural next step is to combine these two contributions by incorporating vendor satisfaction objectives into the simulation-based framework. This would require translating allocation targets, penalties, and vendor performance indicators into metrics that can be computed from operational data, used as policy features, and evaluated under realistic scenarios. Such an integration would allow managers to quantify not only the theoretical cost of vendor satisfaction, but also its behavior under real inventory dynamics, imperfect data, operational constraints, and existing business rules.

A fourth direction concerns policy representation and learning. The thesis compares several policy classes in the marketplace model and studies interpretable score-based policies in the DSF. Future work could explore hybrid policy representations that combine the strengths of these approaches. For example, score-based policies could incorporate terms inspired by opportunity-cost or value-function models, allowing interpretable rules to approximate some of the non-myopic logic of cost-function policies while remaining compatible with existing business systems. Learning-based policies could also be constrained, distilled, or approximated to remain interpretable, for example, through decision-tree policy representations or other symbolic structures (Bastani et al., 2018).

A fifth direction concerns scenario generation, robustness, and deployment validation. Chapter 4 demonstrates that simulation can be used not only to evaluate average policy performance, but also to stress-test policies under alternative cost and inventory conditions. Beyond offline testing, future work should also consider live pilots, A/B tests, or

shadow-mode deployments, where new allocation policies are evaluated against production decisions without immediately affecting customers or vendors.

Overall, future work can build on the foundations developed here by studying richer OAOR settings, better incorporating broader stakeholder objectives, creating hybrid allocation approaches that utilize the advantages of distinct methods, and validating allocation systems in live or semi-live environments. In this way, order allocation can continue to develop as both an analytically grounded research field and a practically relevant decision-support problem.

Bibliography

P. Amorim, H. Meyr, C. Almeder, and B. Almada-Lobo. Managing perishability in production-distribution planning: a discussion and review. *Flexible Services and Manufacturing Journal*, 25:389–413, 2013. doi: 10.1007/S10696-011-9122-3.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. *CoRR*, abs/1805.08328, 2018. URL http://arxiv.org/abs/1805.08328.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

Alexander Hübner, Heinrich Kuhn, and Johannes Wollenburg. Last mile fulfilment and distribution in omni-channel grocery retailing: A strategic planning framework. *International Journal of Retail and Distribution Management*, 44(3):228–247, 2016. doi: 10.1108/IJRDM-11-2014-0154.